DESIGN, ANALYSIS, AND EXPERIMENT ON MULTIPLE SERVERS TECHNOLOGY FOR VIDEO-ON-DEMAND SERVICE IN DISTRIBUTED NETWORKS

CHEN, LONG

NATIONAL UNIVERSITY OF SINGAPORE

2003

DESIGN, ANALYSIS, AND EXPERIMENT ON MULTIPLE SERVERS TECHNOLOGY FOR VIDEO-ON-DEMAND SERVICE IN DISTRIBUTED NETWORKS

CHEN, LONG

(*M.Eng.& B.Eng., NWPU, P.R.China*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

I would like to express my deepest appreciation to Prof. Bharadwaj Veeravalli for his inspiration, excellent guidance, support and encouragement. His erudite knowledge, the deepest insights have been the most inspiration and made this research work a rewarding experience. I owe an immense debt of gratitude to him for not only giving me the invaluable guidance and support about this research work, but also taking care of my life in Singapore. His rigorous scientific approach and endless enthusiasm have influenced me greatly. Without his kindest help, this thesis and many other works would have been impossible.

I sincerely acknowledge all the help from all members in Open Source Software Laboratory, the National University of Singapore. Their kind assistance and friendship have made my life in Singapore easy and colorful.

Thanks also go to the faculties in the Department of Electrical & Computer Engineering, the National University of Singapore, for their constant encouragement and valuable advice.

Acknowledgement is extended to National University of Singapore for awarding me the research scholarship and providing me the research facilities and challenging environment during my study time.

Last but not least, I would thank my family members for their support, understanding, patience and love during this process of my pursuit of a M.Eng, especially to my girlfriend, Liu Yu. This thesis, thereupon, is dedicated to them for their infinite love.

Contents

List of Figures

List of Tables

Summary

In this thesis, a generalized approach is proposed to retrieve a long-duration movie requested using a network based Video-on-Demand(VoD) service infrastructure employing multiple servers. In this multi-server environment, a *play-while-retrieve* (PWR) *playback strategy* is designed and analyzed such that the access time (waiting time for the clients) is minimized. For this strategy, both the *single installment* and *multi-installment* retrieval strategies are used to analyze the performance of the service system. For the above mentioned retrieval strategies, the closed-form expressions for a minimum access time are explicitly derived. For the case of multi-installment retrieval strategy, a asymptotic performance analysis is conducted to quantify the ultimate performance bounds of the strategy. The impact of a large scale network is analytically demonstrated as well as the impact of indefinitely increasing the number of installments on the performance of such a multi-server service system. Then the problem of buffer management at the client site is addressed, which is a closely related issue that has a significant influence on the performance of the strategy and also serves as a key issue in making the service system attractive for clients. The minimum amount of buffer expected at the client site is rigourously derived to have a smooth presentation with this multi-server service structure. In the simulation experiments, the performance of PWR strategy is compared with that of *play-after-retrieve* (PAR) strategy. Further, the PWR strategy is implemented in a real-life VoD system to evaluate the applicability and performance.

Chapter 1

Introduction

Recent advances in information storage, retrieval, and communication have thrown up many novel concepts with possible potential for commercial exploitation. VoD service is one such application which had generated a certain amount of interest because of its potential use in the distributed system. It is a service that enables users to select a video (or a movie) from a large collection, while sitting at individual places, and have a good control over the viewing of the video as when using a conventional VCR.

1.1 Introduction to VoD Service

A typical VoD system consists of a video storage server, a high-speed network, clients with a control device having VCR-like function, in front of a video terminal or a display device. The display device is connected through a network interface, called a set-top-box (STB), to the local access network which, in turn, connects to the backbone high-speed network to which is connected a local video storage server [1]. This entire system is shown in Figure 1.1. High-speed connection channels are used so that the video stream can be supported. The backbone network delivers multiple video streams from the switch at the server end to the switch at the user end. The STB is the customer's interface device between the display unit and the

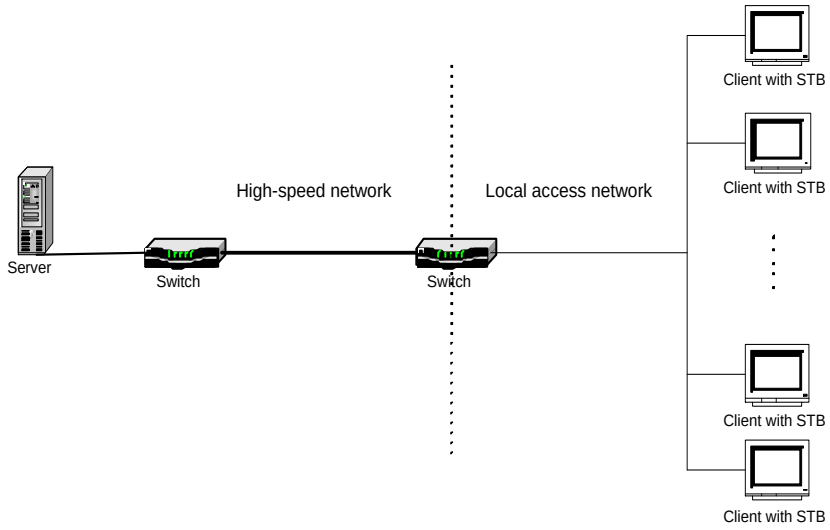

Figure 1.1: Architecture of a typical VoD system

network. It is likely to develop into a powerful processor in the near future and serve a variety of functions like processing customer requests and sending it to the server, receiving, storing and decoding video and other data, and displaying it consistently on the display unit.

Apart from its entertainment applications, VoD technology in its initial phases held out the promise of becoming a major medium of distributed services such as *remote-educating, video conference*, and so on. However, in spite of the falling prices of communication and related hardware, the bandwidth and storage requirements were of such high order that it hindered commercial exploitation of VoD. The large initial investments required to lay out high capacity communication lines and provide other infrastructural support, in places where already communication lines of lower capacity (but sufficient for supporting conventional TV and telephone traffic) existed, became a serious impediment. Another reason was that the basic advantage of VCR-like interactivity which an ideal VoD service promised (but in reality, few services did), did not prove to be of much attraction when compared to the flexibility of channel changing available in multiple channel TV broadcasts. However, it would be a mistake to underestimate the future potential of VoD services. The basic idea behind VoD can still be exploited in virgin territories where communication infrastructure is not yet fully developed.

Even apart from this the Internet information revolution is bound to push up the demand for higher capacity communication channels to homes even in well-developed territories [1]. The combination of the Internet and VoD may very well be the basis for entertainment, business, and education of the future. This appears to be feasible from the recent trends in residential broadband internet services [2].

1.2 Scheduling Strategy of VoD Service

Although the professed aim of a VoD system is to supply all available movies to all clients at any time, in reality few VoD systems can provide such high level service. The limitations are due to the bound on the number of sessions that can be supported by a particular device type, the limitations of the communication network bandwidth, availability of a movie demanded by the consumer, storage limitations, and so on. Thus, to resolve these problems, different scheduling strategies are employed in VoD systems from different viewpoints.

1.2.1 Objectives of the Scheduling Strategy

A major issues in making VoD a competitive and cost-effective service is the efficient utilization of resources, especially the bandwidth and storage capacity. It is likely to remain a major issue even when better resources become available in the future. On the other hand, because of the large volume of data involved in the process and stringent continuity and real-time constraints, the VoD service poses challenges that are different from the standard file transfer operations in the network. The necessity of efficient usage of scarce resources like network bandwidth and server capacity (in terms of I/O bandwidth) demands novel and easy-to-use schemes for scheduling continuous video streams.

To maintain a continuous delivery of streams, the resources at the server and at the network must be carefully utilized. Even though the backbone high-speed network has a large band-

width it can support only a finite number of streams. The resource bottleneck at the server is the I/O disk bandwidth which depends mainly on the number of disks in use. The disk is a mechanical system and so its access time is slower by several orders of magnitude than the electronic components. This imposes a hard limit on the number of I/O streams supported by the disk.

Apart from good scalability the other related objectives of a scheduling strategy are that of performance in terms of reduction in I/O bandwidth demand, reduction in customer waiting times, fairness in providing service to all requests, and good response to interactive user operations. Moreover, scheduling strategies for video streams in VoD systems have to take into account certain non-standard factors like the uneven distribution of requests for movies (that is, a handful of popular movies get the most number of requests whereas others get very few), existence of well-defined prime times (normally during evenings and late nights during weekends), large number of customers spread over a large geographical area serviced by a metropolitan or wide area network, the necessity to maintain a certain quality of service (QoS) level at the user end, and lastly, the exercise of VCR-like controls by the customers.

1.2.2 Scheduling Strategies on Multiple Servers System

As shown in Figure 1.1, the common architecture shared by most VoD system service providers is a single server model. However, the issues such as scalability (in terms of growing client population) and server system fault tolerance can not be satisfactorily addressed by using this single server architecture. The issues mentioned above are essentially due to the fundamental limitations of the single server architecture. Due to this reason, in recent years, researchers have begun to investigate video server designs and implementations based on employing multiple servers architecture to provide high-quality, highly scalable, fault-tolerant service on networks. In this thesis, a multi-server architecture in which several geographically separated high-end multimedia servers can co-operate and share the available movie files in order to

serve their local subscribers is considered. This multi-server service structure is particularly attractive because of following reasons.

Firstly, on a network-based service rendering environment, if a single server system is used, however sophisticated it may be (in terms of speed and capacity), there is a continuous "work pressure" that is enforced on the system by the demands of large client population. For instance, when there is a continuous demand for a long duration video retrieval by several clients, more than 80% of the time is spent in servicing these requests, while some small number of requests demanding short services may undergo long waiting times. By employing multi-server strategy, the work pressure can be balanced among the servers.

Secondly, by using multi-server strategy, even low-bandwidth servers that may not be efficient to utilize can now be a part of several servers in retrieving the movie.

Thirdly, considering fault-tolerance aspect, even under server/link failures, the workload imbalance can be gracefully taken care by the remaining servers. Since multiple servers are engaged in the retrieval process, failure of one or more servers will allow the service to continue without any interruption so long as there is at least one server to cater. In contrast, with the conventional system, most probably the clients may need to be rescheduled, or the presentation gets affected. Also, as shown in the rigorous simulation study in [3, 4], scalability of the physical system and heterogeneity of the system can be easily accounted in the design, as the size of the portions retrieved from each of the servers depends on the available connection bandwidth and playback rate of the movie. Finally, from service provider's perspective, since each server is engaged only for a short while in retrieving a portion, the number of clients that can be entertained can be maximized.

In fact, the retrieval model proposed in [5] is very close to the model adopted in [4] and in this thesis. In the former model, a single video is distributed over multiple servers, whereas each server only stores a subset of the original video data. To retrieve a video, all servers storing pieces of the requested video must send their subset in a coordinated fashion to the client.

Our model differs from this model in several ways. Firstly, in the former model, each server only stores a subset of the original video data. Hence, the client cannot watch certain movie if any server storing this movie is offline. In contrast, this scenario will not happen in the latter model unless all movie servers storing this movie are offline because the entire movie is available with all (or a subset of all) servers. Secondly, in the former model, the distribution of the subset of the video data must be re-organized once after adding the movie server to take advantage of the added bandwidth and capacity. The servers in the later model can be simply added to the current working system, without re-configuring other servers. Then, the system can scale up easily. Finally, in the former model, all servers contribute an equal share to the overall effort of retrieving the video. This scheduling does not take account of the differences among movie servers, such as affordable bandwidth, current load, and so on. These factors have been explicitly taken into account in the design of our system.

Recently, the concepts of using a *concurrent push* [6] and *pull-based* [7] parallel server design are proposed. In the concurrent push based system, also referred to as a *server-push* scheme, the retrieval of video data is carried out completely by the server. That is, the server completely schedules the retrieval and transmission of the data throughout the session [6]. In the case of *client-pull* scheme, the client holds the responsibility to send requests periodically to the server to retrieve the data. Essentially, the server is considered as a "stateless" entity without explicitly keeping track of the customer throughout the session. Besides these two approaches, there is a third approach that incorporates proxy at the client site [3]. With this scheme, a proxy requests the servers to send data directly to the proxy which is located at client machine. After the data is processed by the proxy, the data directly goes to the client application, thus avoiding further network communications. In a pull-based design, it was shown in [7], the need for inter-server synchronization is completely eliminated and also by a careful design of admission control algorithm, the loads across the serves are carefully balanced. This in turn facilitates the system to scale-up in a linear fashion. Several perfor-

mance measures such as client buffer requirements, pre-fetch delays, and scheduling delays are considered in this study. Striping techniques, general approaches for distributing data over multiple devices, are introduced in [3]. In the case of *time striping* methodology, a video data is striped across multiple servers in units of video frames, each unit is of fixed length (measured in time). If a stripe unit has L frames, each server will stream L frames per second and with N servers on the whole, each round with a time span of (NL/F) seconds, L frames will be retrieved from each server and delivered to the client, where F is a constant playback rate. In the case of *space striping*, the video data is divided into fixed-size (measured in bytes) stripe units. This is done in the view of optimizing and simplifying the storage requirements and buffer management at the servers. This strategy is also employed in [8, 9].

While both the studies in [3, 6, 7] and the model adopted in the design of strategy in [4] attempt to use more than one server to serve requests, there are some significant differences between them. The main difference between these studies is that in [3, 6, 7], the video data is basically partitioned and stored (stripped) across many servers, and hence, each server can render only the portions that are stored with it. Also, the data that is stored across these servers are of uniform sized stripes. However, work reported in [4] assumes that the entire movie is available at some servers (just as in conventional video rental stores) and each of these servers is *scheduled* to retrieve only a portion of the movie. Also, this strategy allows non-uniform sized portions to be retrieved from the servers in view of minimizing the access time on a highly delay-sensitive networked environment. Thus, the model in the latter study explicitly accounts any non-zero communication delays in the process of minimizing the access time, while the studies in [3, 6, 7] are more suited when communication delays are negligible. In fact, [10] extends and generalizes the treatment followed in [4] for the case of multiple servers and multiple clients and also carries out load balancing across the servers.

1.2.3 Related Research on VoD Service

Now we present some of the closely related works from the existing literature. In the case of VoD services, depending on the popularity of the movie, the per user cost can be decreased when clever placement of movies on the network is carried out [11]. In the literature, the designs of a VoD system employing several technologies, ranging from disk array technology to sophisticated scheduling policies are realized to optimize several performance measures of interest. Typically, these measures could be to maximize the number of clients that can be supported (admission control algorithms [12]), to minimize the waiting time (access time) of the users [13, 14, 15, 16], or to efficiently use the available buffer space [17], to quote a few.

Apart from the recent attempts employing multiple servers in the literature mentioned in Section 1.2.2, a lot of researches have been devoted in designing several retrieval scheduling strategies, e.g., retrieval scheduling for disk (single and arrays of disks)[18, 19, 20] and tape cartridge [21] for minimizing the access time of a requested document, while maximizing the number of continuous streams that can be supported and minimizing the buffer requirements at the client site. These scheduling strategies are tightly related with the types of storage media. Besides, there are scheduling strategies not related to storage media, e.g., [22] assigns requests to servers with lighter load (load balancing ability) to maximize retrieval capacity. Also, there are scheduling methods devised exclusively for broadcasting. These include, the pyramid broadcasting [13], permutation-based pyramid broadcasting [14] and skyscraper broadcasting [23]. In these schemes, basically, the idea is to partition each movie into several segments and broadcast them periodically, towards a goal of achieving a minimum access time. However, broadcasting schemes have inherent disadvantages of making the client wait through the entire broadcast batch to get his/her choice [1] and usually require high capabilities of client-end configuration.

The problem of data organization and storage is well studied in the literature [24, 25, 26].

Some studies also focus on the design of movie buffer caching strategies for effectively utilizing the memory and reducing disk I/O overheads [27] and dynamic network resource allocation for improving transmission rates with low jitter variation in media streams [28]. In [28], a dynamic bandwidth management policy that uses the concept of TDMA is proposed and its performance is evaluated. The authors also propose a scheme that allows a graceful degradation of the QoS, when the underlying LANs capacity is not sufficient to meet the total demand. These studies essentially deal with the data layout problems for easy and high speed access from a single disk or an array of disks (RAID technology). On the other hand, problems that deal with the provision of services are classified into three types, namely *data-centered*, *user-centered*, and *hybrid* [1]. Conventional broadcasting [1] and a recently proposed pyramid broadcasting [13, 14] are the examples of the data-centered approach. A very recent work in [17] considers employing multiple servers to retrieve multimedia objects, but from a different objective. In this work, the authors design a scheduling scheme, referred to as an *application layer broker*(ALB), at the client site. Typically, a client negotiates with a group of servers and identifies the best server to retrieve an object. This scheme attempts to minimize the buffer space requirements at the client site.

1.3 Main Contributions of the Thesis

In this thesis, efficient movie retrieval strategies that minimizes the access times of movies by clients employing this multi-server distributed architecture are designed and analyzed. The main contributions are listed below.

- A *play-while-retrieve* (PWR) playback strategy for a multi-server environment is designed and analyzed. For this strategy, both the single installment and multi-installment retrieval strategies are used to analyze the performance of the system.

- For the above mentioned retrieval strategies, closed-form expressions for a minimum

access time are explicitly derive.

- For the case of multi-installment retrieval strategy, since the retrieval follows several rounds of installments, the ultimate performance bounds (asymptotic performance analysis) that quantify the limiting performance of our strategy are derived. Further, the impact of a large scale network as well as the impact of indefinitely increasing the number of installments with the PWR strategy are demonstrated, thus quantifying the performance of such a multi-server service architecture.

- The problem of buffer management at the client site is addressed, which is one of the closely related issues that has a significant influence on the performance of the strategy. Relationships that quantify the minimum amount of buffer expected at the client site to has a smooth presentation with this multi-server service structure are derived, for both the single installment and multi-installment retrieval strategies.

- Finally, to testify all theoretical findings, simulation experiments and implementation are conducted. In the experiments, the performance of PWR strategy is compared with that of PAR strategy and certain important points that are crucial for implementing a real-life working multi-server service system are discussed.

1.4 Organization of the Thesis

The thesis is organized as follows.

In this chapter, we first give out the basic concept of the VoD service. Then, the problem of the scheduling strategy of VoD service and other related works are described.

In Chapter 2, we present the problem setting and introduce the necessary notions and terminologies used throughout the thesis. We also describe the basic idea behind the playback and retrieval strategies.

In Chapter 3, we present the design and analysis of the PWR strategy to minimize the access time. For this strategy, both the single installment and multi-installment retrieval strategies are used to analyze the performance of the service system.

In Chapter 4, we present a rigorous analysis on the buffer management at the client site using the PWR strategy. The minimum amount of buffer expected at the client site is rigourously derived to have a smooth presentation with this multi-server service structure.

In Chapter 5, we present our simulation experiments. The performance of the PWR strategy is compared with that of the PAR strategy.

In Chapter 6, we present the implementation of our strategy in a real-life VoD system. The retrieval process and the performance are described.

In Chapter 7, we conclude the thesis with some open-ended issues to be addressed.

Chapter 2

System Model and Preliminary Remarks

In this chapter, we present the problem of retrieval strategy on the multi-server system more formally, describe the network architecture that is considered, and introduce the necessary definitions, notations and terminologies. We envisage the underlying network as shown in Figure 2.1. In the network architecture shown, each server serves its respective local customers and customers situated at other sites. The request for viewing a movie is individually initiated by local customers/clients on each server. Upon an arrival of a request, the server seeks the requested movie locally first. If this movie is available locally, then the movie is retrieved and presented to the user at once. However, if the requested movie is not available locally, this original server can obtain the information about the requested movie on other servers by employing look up services, such as the directory service. Then the requested movie can be retrieved from one or more servers employing our proposed strategy. In the following, we describe the basic retrieval mechanism employed in our strategy.

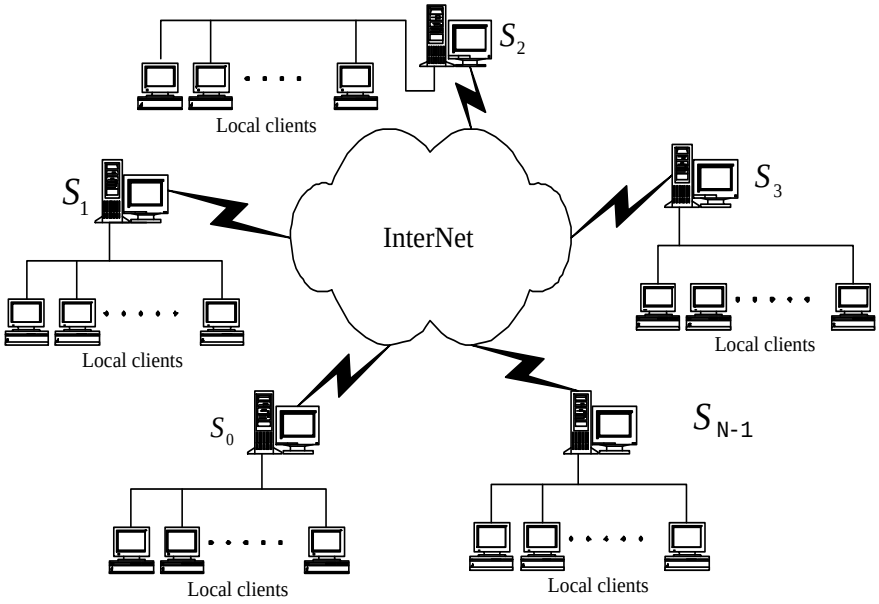

Figure 2.1: Architecture of a multi-server VoD system

2.1 Description of the PWR Strategy

We now describe the PWR strategy used in this thesis. Consider a scenario in which a requested movie is not available locally at the original server, denoted as, S. Without loss of generality, we assume that the requested movie is present at servers S_0, S_1 and S_2. Let the total size of the requested movie be L, measured in *bits*. The connection bandwidths of channels from other servers(in this case S_0, S_1 and S_2) to the local server(in this case server S) are denoted as $bw_i,\ i = 0, 1, 2$, measured in *bits per second*. Let the playback rate at the client site be R_p, measured in *bits per second (bps)*. Once locating the respective servers having the requested movie, server S adopts the following strategy. From each server a portion of the entire movie, denoted as $m_i,\ i = 0, 1, 2$, is retrieved and is collected by S in a particular order. Upon receiving the first portion of the movie from S_0, the playback may start at the user terminal, when retrievals from other servers are underway.

As mentioned in Chapter 1 , presentation continuity is one of the QoS requirements for a multimedia presentation. Thus, in order to start the playback when retrievals from other

servers are underway, the size of the portion retrieved must be such that there should not be any data starvation for playback. In other words, the size of the portion retrieved must guarantee the presentation continuity. Now, the retrieval strategy must be such that before the playback of the first portion (retrieved from S_0) comes to an end, next portion of the requested movie data should be made available from S_1. This retrieval process continues until all the movie is retrieved from the set of servers.

The above example describes our retrieval strategy in which server S only retrieves one portion of the movie from each server. This strategy is referred to as *single installment* retrieval strategy. On the other hand, the retrieval process may be such that server S may retrieve movie portions from each server in *multiple installments*. Thus, each server participates in the retrieval process more than once. We will thoroughly describe these strategies in Chapter 3.

In [4], PAR strategy was attempted. In this strategy, a client is allowed to start the playback only after the client has received the entire portion from the first server(S_0, in our example above). However, in our PWR strategy, we relax this assumption and design a strategy which allows an *early start* of the playback which guarantees a presentation continuity. Thus, as soon as the *critical size* of first portion has been retrieved, the playback can be initiated on the client site. In other words, the client can start playing this portion while the remaining portion is being retrieved and hence the name of *PWR*. The critical size, denoted as $cs_i,\ i = 0, 1, 2$, is the minimum size of movie that a client should retrieve before the playback of this portion could be started so as to avoid data starvation during playback. This critical size is indeed dependent on the available connection bandwidth of the channels and the playback rate of the movie at the client site. For every portion that is to be retrieved from a server, our strategy recommends a critical size that should be retrieved in order to avoid data starvation. Figure 2.2 shows the whole process of above example.

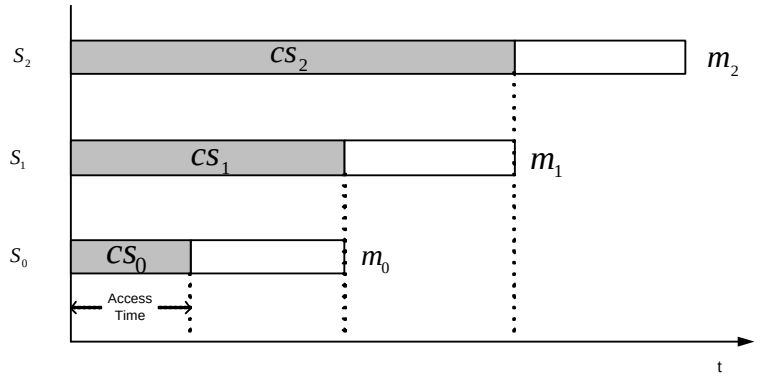

Figure 2.2: Example of the PWR strategy

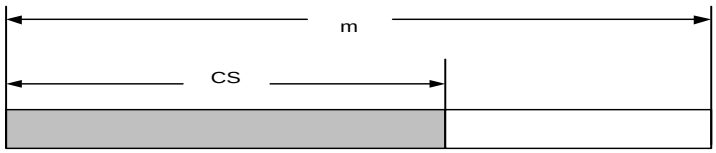

Figure 2.3: Determination of the Critical Size

2.1.1 Determination of Critical Size

We now describe how this critical size can be computed. This will be used later in the analysis of our strategy. Consider a scenario in which a portion of the movie of size m is to be retrieved from a server using a connection bandwidth of bw demanding a playback rate of R_p at the client site. The client can safely start playing the portion after the critical size cs of this portion has been retrieved. Figure 2.3 shows this concept. From the figure, we observe that in order to guarantee a continuous playback, the time to retrieve the remaining portion $(m - cs)$ must be not greater than the entire playback duration of the portion m. In other words,

$$\frac{m}{R_p} \geq \frac{m - cs}{bw} \tag{2.1}$$

Thus, by satisfying this condition (2.1), we ensure that the retrieval of the remaining portion will not affect the continuity of the playback at any time instant. Further, when sufficiently large amount of bandwidth is available (high bandwidth networks), i.e., whenever $bw \geq R_p$, we note that the playback can almost start instantaneously, which is consistent with our

strategy. This means that we avoid buffering the data. However, in reality, most remote VoD servers other than the original server cannot support a client with a connection bandwidth that is higher than the playback rate, for example, 1.5Mbps for MPEG1 movie file. Hence, without loss of generality, we assume $R_p > bw$ throughout the entire thesis. Further, from practical perspective, we have,

$$cs \geq max\{\frac{(R_p - bw)m}{R_p}, \delta\} \tag{2.2}$$

where parameter δ is the minimum size that a video player needs to initiate a playback and this value depends on different players. Note that when compared to the critical size used in (2.1), this parameter δ has a different interpretation. Critical size is used to guarantee a continuous presentation and is determined by our retrieval strategy, while δ is a parameter that is associated with the techniques of players, wherein each player "expects" a minimum amount of data to kick-start the presentation process and is completely unaware and independent of continuity in presentation. Without loss of generality, we use $\delta = 0$ throughout the thesis. Thus, as soon as cs of one portion has been retrieved, we can start playing the portion safely and the rest of the portion can be retrieved continuously while the playback is underway. From (2.2), we can see that the critical size bears a linear relationship with the movie size for a given bandwidth and playback rate.

2.2 Some Definitions

Throughout the thesis we use the following definitions.

1.*Retrieval schedule distribution*: This is defined as an N ordered tuple m given by,

$$m = (m_0, m_1, \ldots, m_{N-1}) \tag{2.3}$$

where m_i is the portion of the movie retrieved from server $S_i, i = 0, 1, 2, \ldots, N-1$. Further,

$$\sum_{k=0}^{N-1} m_k = L \tag{2.4}$$

and

$$0 \leq m_i \leq L, \qquad i = 0, 1, 2, \ldots, N-1 \tag{2.5}$$

The set of all such retrieval schedule distributions is denoted as Γ.

2.*Critical Size distribution*: This is defined as an N ordered tuple cs given by

$$cs = (cs_0, cs_1, \ldots, cs_{N-1}) \tag{2.6}$$

where cs_i is the critical size of corresponding portion m_i, $i = 0, 1, 2, \ldots, N-1$. Using equality relation in (2.2), we have,

$$cs_i = \frac{(R_p - bw_i)m_i}{R_p} \tag{2.7}$$

3.*Access Time*: This is defined as the time between the instant at which the servers start uploading their portions to the instant at which the presentation starts. This is denoted as, $AT(m)$. According to our scheme motioned before, this is the time to access the critical size of the first portion of the requested movie, given by cs_0/bw_0, where bw_0 is the connection bandwidth of the established communication channel from S_0 to S (supposing the first portion is retrieved from S_0).

4.*Minimum access time*: This is defined as ,

$$AT^*(m^*) = min_{m \in \Gamma} AT(m) \tag{2.8}$$

where, $m^* = (m_0^*, ..., m_{N-1}^*) \in \Gamma$ denotes an optimal retrieval schedule distribution of the entire movie.

Thus, from above set of definitions and the strategy, our objective is to minimize the access time by determining the optimal sizes of the portions of the movie to be retrieved from different servers involved in the retrieval process, which will be presented in the next chapter.

Chapter 3

Design of Movie Retrieval Strategies

In this chapter, we shall present our single installment and multi-installment retrieval strategies in detail and determine the optimal sizes of various portions retrieved from all the N servers in order to achieve the minimum access time.

3.1 Single Installment Strategy

In this strategy, following an order of retrieval, say from S_0 to S_{N-1}, portions of movie are retrieved. Each server participates in the retrieval process only once for a client and hence the name *single installment* strategy. We now derive a closed-form solution for the minimum access time following this strategy. In Figure 3.1 we show the retrieval process using a directed flow graph comprising communication nodes (retrieval) and playback nodes. The arrows capture the precedence relationships in the retrieval and playback portions. For example, portion i can be played after portion $(i-1)$ and after receiving its critical size. Note that the weights of the nodes are indeed the communication and playback durations of the respective nodes. From this figure, we can derive a relationship between the retrieval of portion i and $(i+1)$ and the playback time of the portion m_i with the use of causal precedence

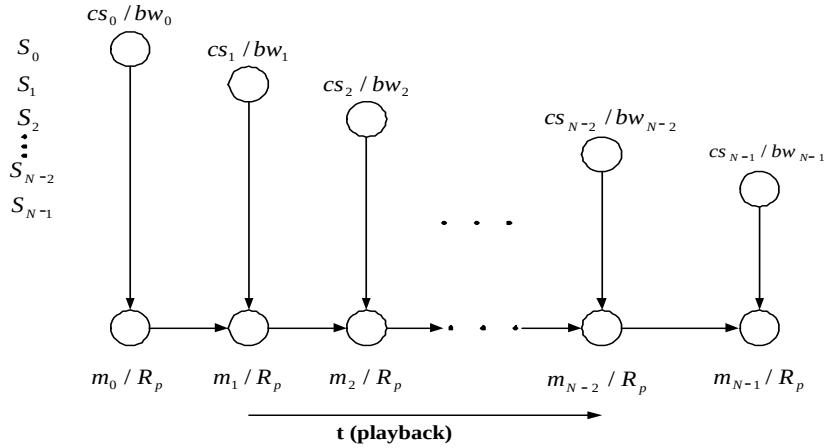

Figure 3.1: Directed flow graph representation using single installment strategy for movie retrieval from N servers

relation and continuity constraint as,

$$\frac{cs_i}{bw_i} + \frac{m_i}{R_p} \geq \frac{cs_{i+1}}{bw_{i+1}} \tag{3.1}$$

By using (2.7) in (3.1), we can have,

$$\frac{(R_p - bw_i)m_i}{R_p bw_i} + \frac{m_i}{R_p} \geq \frac{(R_p - bw_{i+1})m_{i+1}}{R_p bw_{i+1}} \tag{3.2}$$

Further

$$m_{i+1} \leq \frac{R_p bw_{i+1} m_i}{(R_p - bw_{i+1})bw_i} \tag{3.3}$$

Let us denote $R_p bw_{i+1}/(R_p - bw_{i+1})bw_i = \rho_i$. Rewriting (3.3),we have,

$$m_{i+1} \leq m_i \rho_i, \quad i = 0, 1, \ldots, N-2. \tag{3.4}$$

Then, the above set of equations represents a set of recursive equations that can be solved under equality conditions. Note that the use of equality relationships in (3.3) and (3.4) results in the *maximum* size of all the portions other than m_0. Hence, using (2.4) we obtain a *minimum* value for m_0, equivalently the minimum cs_0. In other words, we obtain a minimum access time. Thus, we have a recursive set of $(N-1)$ equations with equality relations from

(3.4). Each m_i can be expressed in terms of m_0 as ,

$$m_i = m_0 \prod_{k=0}^{i-1} \rho_k, \quad i = 1, 2, \ldots, N-1. \tag{3.5}$$

Thus, the above set of $(N-1)$ equations given by (3.5) together with (2.4) are solved to obtain the individual disjoint portions of the requested movie. Substituting each m_i from (3.5) into (2.4), we obtain,

$$m_0 = \frac{L}{1 + \sum_{p=1}^{N-1} \prod_{k=0}^{p-1} \rho_k} \tag{3.6}$$

Substituting (3.6) in (3.5), we obtain the individual sizes of the portions as,

$$m_i = \frac{L \prod_{k=0}^{i-1} \rho_k}{1 + \sum_{p=1}^{N-1} \prod_{k=0}^{p-1} \rho_k}, \quad i = 1, 2, \ldots, N-1. \tag{3.7}$$

Thus, the access time is given by,

$$AT^*(m^*) = \frac{cs_0}{bw_0} = \frac{(R_p - bw_0)m_0}{R_p bw_0} = \frac{L(\frac{1}{bw_0} - \frac{1}{R_p})}{(1 + \sum_{p=1}^{N-1} \prod_{k=0}^{p-1} \rho_k)} \tag{3.8}$$

It may be noted that only when $R_p > bw_0$ our strategy becomes meaningful. This is because of the fact that in the case of high bandwidth connections (more than the playback demand), employing a pool of servers to retrieve a movie results in insignificant, if not, no gain in access time.

3.1.1 Homogeneous Channels

We consider a network with identical connection bandwidths among servers, i.e., $bw_i = bw$, for all $i = 0, 1, ..., N-1$. In this case, using (3.6) and (3.7), the individual sizes of the portions retrieved from the servers S_i are given by,

$$m_0 = \frac{L(\rho - 1)}{\rho^N - 1} \tag{3.9}$$

$$m_i = \frac{L(\rho-1)\rho^i}{\rho^N - 1}, \quad i = 1, 2, \ldots, N-1. \tag{3.10}$$

Hence, the access time is given by,

$$AT^*(m^*) = \frac{cs_0}{bw} = \frac{L(\rho-1)}{\rho(\rho^N-1)bw} \tag{3.11}$$

3.1.2 Effect of Sequencing

The strategy described above assumes that the retrieval follows a *fixed sequence*, i.e., from S_0 to S_{N-1}. However, it may be noted that given a set of N servers, we have $N!$ retrieval sequences possible. Following the steps described in [4], even for our PWR single installment strategy described in this thesis, we can use the following lemma and theorem to prove that the access time remains independent of the retrieval sequence.

Lemma 1. *Let the access time of a requested movie file by the server S be denoted as $AT(m, \sigma(k, k+1))$, where $\sigma(k, k+1) = (S_0, S_1, ..., S_{k-1}, S_k, S_{k+1}, S_{k+2}, ...S_{N-1})$, denotes the sequence in which the requested movie file is retrieved from the servers. Then, for a sequence $\sigma'(k, k+1) = (S_0, S_1, ..., S_{k-1}, S_{k+1}, S_k, S_{k+2}, ...S_{N-1})$, the access time $AT(m', \sigma'(k, k+1))$ is equal to $AT(m, \sigma(k, k+1))$ where, $\sigma'(k, k+1)$ denotes a retrieval sequence in which the adjacent channels k and $k+1$ are swapped, i.e., portion from server S_{k+1} is retrieved first and then from server S_k.*

Proof: The denominator of (3.8) can be written as :

$$denom(m) = 1 + \rho_0 + \rho_0\rho_1 + \rho_0\rho_1\rho_2 + ... = 1 + \rho_0(1 + \rho_1(1 + \rho_2(...))) \tag{3.12}$$

We can distinguish two cases depending on whether the first server S_0 is involved or not. If S_0 is not involved, when a switch is made between two successive servers, the new denominator is different from the original one in three ρ terms. This difference can be written as

$$denom(m') - denom(m) = \sum_{j=0}^{i-1} \rho_j$$

$$[\frac{R_p bw_{i+1}}{(R_p - bw_{i+1})bw_{i-1}}(1 + \frac{R_p bw_i}{(R_p - bw_i)bw_{i+1}}(1 + \frac{R_p bw_{i+2}}{(R_p - bw_{i+2})bw_i}(1 + \rho_{i+3}(...))))$$

$$-\frac{R_p bw_i}{(R_p - bw_i)bw_{i-1}}(1 + \frac{R_p bw_{i+1}}{(R_p - bw_{i+1})bw_i}(1 + \frac{R_p bw_{i+2}}{(R_p - bw_{i+2})bw_{i+1}}(1 + \rho_{i+3}(...))))] \quad (3.13)$$

With little algebraic manipulation, the above equation returns zero.

In the second case, where the first two servers S_0 and S_1 are switched, the difference in access time is,

$$AT(m) - AT(m') = \frac{L(\frac{1}{bw_0} - \frac{1}{R_p})}{denom(m)} - \frac{L(\frac{1}{bw_1} - \frac{1}{R_p})}{denom(m')}$$

$$= L\frac{(\frac{1}{bw_0} - \frac{1}{R_p})denom(m') - (\frac{1}{bw_1} - \frac{1}{R_p})denom(m)}{denom(m)denom(m')} \quad (3.14)$$

By using (3.12)($denom(m)$ and $denom(m')$ differ in two ρ-terms, ρ_0 and ρ_1), the numerator of the above fraction becomes equal to

$$(\frac{1}{bw_0} - \frac{1}{R_p})(1 + \frac{R_p bw_0}{(R_p - bw_0)bw_1}(1 + \frac{R_p bw_2}{(R_p - bw_2)bw_0}(1 + \rho_2(...))))$$

$$-(\frac{1}{bw_1} - \frac{1}{R_p})(1 + \frac{R_p bw_1}{(R_p - bw_1)bw_0}(1 + \frac{R_p bw_2}{(R_p - bw_2)bw_1}(1 + \rho_2(...)))) \quad (3.15)$$

which in turn can be immediately proven to be equal to zero.

Therefore, one can easily conclude that the order in which the portions are downloaded only affects the respective size distribution, but not the access time when the retrieval order is changed. We prove this claim in general for the case of N servers, as follows.

Theorem 1. *Given a pool of N video servers capable of rendering the requested movie file, using the PWR single installment strategy, the access time is independent of the retrieval sequence used.*

Proof: One can easily prove the theorem with the aid of Lemma 1. Any valid sequence of servers can be derived from a single sequence by switching the positions between adjacent servers. Lemma 1 guarantees that these operations do not affect access time.

3.2 Multi-installment Strategy

In this section, as opposed to the idea of retrieving the movie portions from each server in one installment, we attempt to design a strategy in which each server takes part in retrieval process in more than one installment. This strategy is referred to as *multi-installment* strategy. Thus, starting from server S_0 to S_{N-1}, the individual portions retrieved in the first installment are, $m_{0,0}, \ldots, m_{N-1,0}$, in the second installment we have, $m_{0,1}, \ldots, m_{N-1,1}$, and so on, until n-th installment, given by, $m_{0,n-1}, \ldots, m_{N-1,n-1}$, respectively. Similar to the single installment strategy, the continuity of the presentation must be guaranteed during playback using this multi-installment strategy.

3.2.1 Recursive Equations and Solution Methodology

Figure 3.2 shows the entire process of this retrieval strategy. Let $m_{i,j}$ represents a portion of the total movie retrieved from server S_i during the j-th installment, where $j = 0, 1, \ldots, n-1$. Thus, there are a total of Nn portions of the movie that are retrieved from servers S_0 to S_{N-1} in n installments. Further,

$$\sum_{i=0}^{N-1} \sum_{j=0}^{n-1} m_{(i,j)} = L \tag{3.16}$$

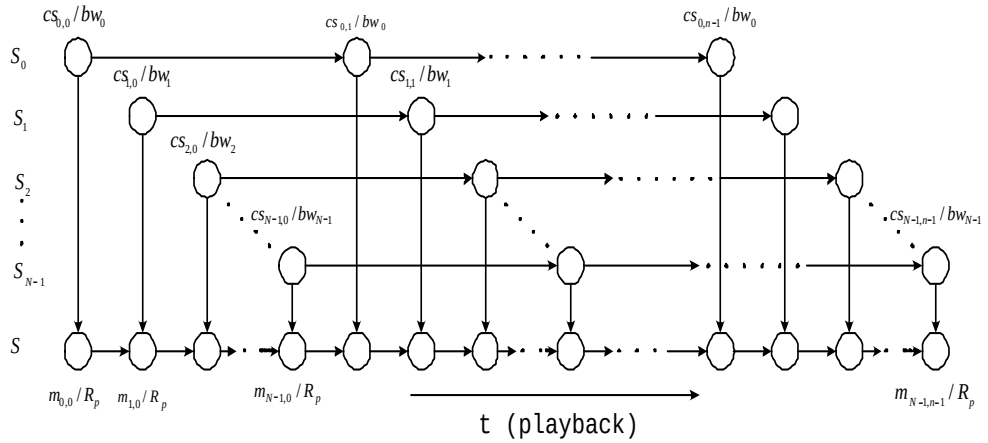

Figure 3.2: Directed flow graph representation using multi-installment strategy for movie retrieval from N servers

Let $cs_{i,j}$ denote the critical size of the corresponding portion $m_{i,j}$. Similar to (2.7), we have,

$$cs_{i,j} = \frac{(R_p - bw_i)m_{i,j}}{R_p} \tag{3.17}$$

It can be deduced from Figure 3.2 that the causal precedence relations and the continuity relationships impose the following inequalities:

$$\frac{cs_{k,0}}{bw_k} + \frac{m_{k,0}}{R_p} \geq \frac{cs_{k+1,0}}{bw_{k+1}}, \quad k = 0, 1, \ldots, N-2. \tag{3.18}$$

For $i = 1, 2, \ldots, n-1$, we have,

$$\frac{\sum_{p=k+1}^{N-1} m_{p,i-1} + \sum_{p=0}^{k-1} m_{p,i}}{R_p} \geq \frac{cs_{k,i}}{bw_i}, \quad k = 0, 1, \ldots, N-1. \tag{3.19}$$

The *minimum* size of $m_{0,0}$, which determines the minimum critical size can be obtained by seeking the *maximization* of all other $m_{i,j}$. This goal can be achieved by using the equality relationships in (3.18) to (3.19) together with (3.16). We obtain,

$$m_{i+1,0} = m_{i,0}\left(\frac{R_p bw_{i+1}}{(R_p - bw_{i+1})bw_i}\right), \quad i = 0, 1, \ldots, N-2. \tag{3.20}$$

For $i = 1, 2, \ldots, n-1$, we have,

$$m_{k,i} = \left(\sum_{p=k+1}^{N-1} m_{p,i-1} + \sum_{p=0}^{k-1} m_{p,i}\right)\frac{bw_k}{R_p - bw_k}, \quad k = 0, 1, \ldots, N-1. \tag{3.21}$$

Since access time is now a function of both the number of servers (N) and the number of installments (n) used, we denote the access time, using multi-installment strategy, as $AT(N, n)$. The access time is given by

$$AT(N, n) = \frac{cs_{0,0}}{bw_0} \tag{3.22}$$

where $cs_{0,0}$ can be obtained by solving the recursive equations (3.20) to (3.21) together with (3.16) above. Note that the complexity of this procedure is $O(Nn)$.

3.2.2 Homogenous Channels

Although the generic case posed above is complex to solve to obtain a closed-form solution, for the case of identical connection bandwidths we attempt to derive an expression for the access time given by the multi-installment strategy. Thus, the above set of recursive equations ((3.20) and (3.21)) can be rewritten as,

$$m_{k,0} = m_{k-1,0}\frac{R_p}{R_p - bw}, \quad k = 1, 2, \ldots, N-1. \tag{3.23}$$

Then, for $i = 1, 2, ..., n-1$, we have,

$$m_{k,i} = \left(\sum_{p=k+1}^{N-1} m_{p,i-1} + \sum_{p=0}^{k-1} m_{p,i}\right)\frac{bw}{R_p - bw}, \quad k = 0, 1, \ldots, N-1. \tag{3.24}$$

Denoting $bw/(R_p - bw)$ as σ, we have,

$$m_{k,0} = m_{k-1,0}(1 + \sigma), \quad k = 1, 2, \ldots, N-1. \tag{3.25}$$

For $i = 1, 2, ..., n-1$, we have,

$$m_{k,i} = \left(\sum_{p=k+1}^{N-1} m_{p,i-1} + \sum_{p=0}^{k-1} m_{p,i}\right)\sigma, \quad k = 0, 1, \ldots, N-1. \tag{3.26}$$

Now, each of the $m_{k,0}, k = 1, 2, ..., N-1$ from (3.25) can be expressed as a function of $m_{0,0}$ as,

$$m_{k,0} = m_{0,0}P(\sigma, k) \tag{3.27}$$

		j							
$m_{i,j}$	k	0	1	2	3	4	5	6	7
0,0	0	1	0	0	0	0	0	0	0
1,0	1	1	1	0	0	0	0	0	0
2,0	2	1	2	1	0	0	0	0	0
3,0	3	1	3	3	1	0	0	0	0
0,1	4	0	3	6	4	1	0	0	0
1,1	5	0	2	8	10	5	1	0	0
2,1	6	0	1	8	17	15	6	1	0
3,1	7	0	0	6	22	31	21	7	1

Table 3.1: Coefficients table for multi-installment strategy

where, $P(\sigma, k) = (1+\sigma)^k$. We define a transformation $k = i(n-1) + jN$ and denote the portions of the movie retrieved from $S_0, S_1, ..., S_{N-1}$ in n installments as Q_k, $k = 0, 1, ..., Nn-1$, where k is as defined above. Thus, with this transformation and using (3.25) and (3.26), we generate the following Table 3.1. We have shown the table for $N = 4$ and $n = 2$ case. The entries in each row of the table are the coefficients of the respective powers of σ. Thus, the maximum number of columns and rows will be 7, i.e., $(Nn-1)$. As an example, $m_{1,1}$ corresponds to the row Q_5, given by (3.26) as $m_{0,0}(2\sigma+8\sigma^2+10\sigma^3+5\sigma^4+\sigma^5)$, and the entries in the table are precisely these coefficients of the various powers of σ. Thus, generalizing this idea, we have the following (boundary) conditions and a recursive definition to generate a particular entry $E(i,j)$ in the table for arbitrary N and n. The boundary conditions that generate entries for the first installment $(n = 0)$ are given by,

$$E(k,0) = 1, \quad \forall k = 0, 1, \ldots, N-1, \tag{3.28}$$

$$E(k,0) = 0, \quad \forall k = N, \ldots, Nn-1, \tag{3.29}$$

$$E(k,j) = 0, \quad \forall j > k, \ and \ k, j = 0, 1, \ldots, Nn-1, \tag{3.30}$$

$$E(k,j) = E(k-1, j-1) + E(k-1, j), \quad \forall k = 1, 2, \ldots, N-1 \tag{3.31}$$

Note that the first entry $E(0,0)$ is always assumed to be equal to 1, for normalization purposes. Now, for the remaining rows, $Q_N, Q_{N+1}, \ldots, Q_{Nn-1}$, we have,

$$E(k,j) = \sum_{p=k-N+1}^{k-1} E(p, j-1), \quad \forall k = N, N+1, \ldots, Nn-1, \quad j = 1, 2, \ldots, Nn-1, \tag{3.32}$$

Thus, in our example, we have for $Q_5 = m_{1,1} = m_{0,0}(E(5,0) + E(5,1)\sigma + ... + E(5,4)\sigma^4 + E(5,5)\sigma^5)$. This is the polynomial shown above. Following this notion, we can write $m_{i,j}$ as,

$$m_{i,j} = Q_k = m_{0,0} \sum_{i=0}^{k} E(k,i)\sigma^i, \quad \forall k = 1, 2, \ldots, Nn-1, \tag{3.33}$$

We have a total of (Nn) unknowns with $(Nn-1)$ equations. As in the Chapter 3.1, together with the normalizing equation,

$$m_{0,0} \sum_{i=0}^{Nn-1} \sum_{j=0}^{i} E(i,j)\sigma^j = L \tag{3.34}$$

we have a total of (Nn) equations to solve for all the unknowns. Note that each of the $m_{i,j}$ can be expressed in terms of $m_{0,0}$, by using (3.34), we obtain,

$$m_{0,0} = \frac{L}{\sum_{i=0}^{Nn-1} \sum_{j=0}^{i} E(i,j)\sigma^j}, \tag{3.35}$$

where, $E(i,j)$ is generated by using (3.28) to (3.32). Thus, given a set of N video servers having identical connection bandwidth, we obtain the optimal sizes of the portions of the movie to be retrieved from each server by using (3.28) to (3.35).

3.2.3 Asymptotic Analysis

From the Chapter 3.2.2, we can obtain the optimal sizes of the portions of the movie to be retrieved from each server. It is of natural interest to examine the impact of using a large number of servers and large number of installments, as both these parameters influence

the performance. It may be noted that the parameter n is software-tunable, and hence the system designers can use this parameter to improve the performance by increasing the number of installments whenever there are fewer servers available for servicing. At the same time, when the system is large, the number of installments that can be used may chosen to be small. Thus, the flexibility of tuning the parameters (N and n) of the system serves as a QoS assurance to the client by the service provider. Finally, we will attempt to derive asymptotic performance bounds when the number of installments tends to be large (infinity) and number of servers in the system are large (theoretically tending to infinity). These bounds serve as invaluable measures in quantifying the performance of the system. That is, we want to obtain:

$$AT(N,\infty) = \lim_{n\to\infty} AT(N,n) \tag{3.36}$$

$$AT(\infty,n) = \lim_{N\to\infty} AT(N,n) \tag{3.37}$$

Using (3.28) through (3.32), we can rewrite (3.34) as,

$$m_{0,0}\left[\sum_{i=0}^{Nn-1} E(i,0) + \sigma \sum_{i=0}^{Nn-1} E(i,1) + \ldots + \sigma^{Nn-1} \sum_{i=0}^{Nn-1} E(i,Nn-1)\right] = L \tag{3.38}$$

which can be written as

$$m_{0,0}\left(\sum_{j=0}^{Nn-1} R(j)\sigma^j\right) = L \tag{3.39}$$

where,

$$R(j) = \sum_{i=0}^{Nn-1} E(i,j) \tag{3.40}$$

and is defined as the coefficient of σ^j in (3.39). In order to evaluate (3.36), we need to compute,

$$\lim_{n\to\infty} m_{0,0} = \lim_{n\to\infty}\left(\frac{L}{\sum_{j=0}^{Nn-1} \sigma^j R(j)}\right) \tag{3.41}$$

From (3.32),

$$\lim_{n\to\infty} R(j) = \sum_{i=0}^{\infty} E(i-1,j-1) + \sum_{i=0}^{\infty} E(i-2,j-1) + \ldots + \sum_{i=0}^{\infty} E(i-N+1,j-1) \tag{3.42}$$

which, upon using (3.28) through (3.31), reduces to,

$$\lim_{n\to\infty} R(j) = (N-1) \lim_{n\to\infty} R(j-1) \tag{3.43}$$

However, from (3.28) we know that,

$$R(0) = N \tag{3.44}$$

Hence,

$$\sum_{j=0}^{\infty} \sigma^j \left(\lim_{n\to\infty} R(j) \right) = N + N(N-1)\sigma + N(N-1)^2\sigma^2 + N(N-1)^3\sigma^3 + \ldots \tag{3.45}$$

The above equation can be further simplified depending on the condition we impose on the factor $(N-1)\sigma$. The following are the two different cases which may arise.

Case 1: $(N-1)\sigma < 1$

For this case, it can be readily seen that (3.45) can be written as

$$\sum_{j=0}^{\infty} \sigma^j \lim_{n\to\infty} R(j) = \frac{N}{1-(N-1)\sigma} \tag{3.46}$$

Thus,

$$\lim_{n\to\infty} m_{0,0} = \frac{(1-(N-1)\sigma)}{N} L \tag{3.47}$$

Case 2: $(N-1)\sigma \geq 1$

For this case, it is obvious that (3.45) will not converge to a finite value. Thus, it can be seen that

$$\lim_{n\to\infty} m_{0,0} = 0 \tag{3.48}$$

Now, we shall evaluate (3.37). From (3.28) through (3.32), we observe that as $N \to \infty$, $\sum_{i=0}^{Nn-1} \sum_{j=0}^{i} E(i,j)\sigma^j \to \infty$. This means that,

$$\lim_{N\to\infty} m_{0,0} = 0 \tag{3.49}$$

Summarizing the results, we have,

$$AT(N,\infty) \;=\; \frac{1-(N-1)\sigma}{(1+\sigma)Nbw} L, \qquad if\ (N-1)\sigma < 1 \tag{3.50}$$

$$= 0, \quad \text{otherwise} \tag{3.51}$$

$$AT(\infty, n) = 0, \tag{3.52}$$

Further, (3.50) can be rewritten as,

$$AT(N, \infty) = \left(\frac{R_p - Nbw}{NbwR_p} \right) L \tag{3.53}$$

Now, when the number of installments is chosen to be sufficiently large and for a given bw, in order to obtain a specified (user defined or guaranteed by the service provider) access time, we can derive the minimum number of servers required as:

$$N_{min} = \left\lceil \frac{R_p L}{(R_p AT + L)bw} \right\rceil \tag{3.54}$$

On the other hand, when N is fixed, the minimum bandwidth needed to achieve a desired access time is given by,

$$bw_{min} = \left\lceil \frac{R_p L}{(R_p AT + L)N} \right\rceil \tag{3.55}$$

So far, we have given out the detailed results of our strategies on the the access time minimization. In the next chapter, we shall present the problem of buffer management at the client site, which is a closely related problem of our strategies.

Chapter 4

Buffer Management at the Client Site

The client's system requirements have always been considered as one of the most important concerns during the design and implementation of the system. The main reason for the concern lies in the fact that any successful commercial application should always assume minimum requirements at the client site for the service to be attractive.

By and large, one of the most important requirements at the client site refers to the minimum amount of buffer size expected. The buffer space includes space to store the incoming stream under normal conditions, the buffer space needed to prevent any underflow and overflow situations under abnormal conditions, and the buffer space for the local media-player[1] to implement its VCR-like control functions other than normal play, such as rewind, fast-forward, pause, stop, etc. In the rest of this chapter, we focus on the size of buffer expected at the client site under normal conditions using our PWR strategy and derive minimum amount of buffer space expected at the client subscribed to this service.

[1]Different media-players may impose different requirements for a smooth playback

4.1 Buffer Occupancy

The size of buffer demanded by the PWR single installment strategy, B_{single}, at the client site can be derived as follows. Since the buffer occupancy is different at different intervals of retrieval durations, we have:

$$B_{single} = \begin{cases} \sum_{i=0}^{N-1} bw_i t, & \text{if } 0 \le t < AT_{single} \\ \sum_{i=0}^{N-1} bw_i t - \alpha R_p(t - AT_{single}), & \text{if } AT_{single} \le t < \frac{m_0}{bw_0} \\ m_0 + \sum_{i=1}^{N-1} bw_i t - \alpha R_p(t - AT_{single}), & \text{if } \frac{m_0}{bw_0} \le t < \frac{m_1}{bw_1} \\ \dots & \\ \sum_{i=0}^{j-1} m_i + \sum_{i=j}^{N-1} bw_i t - \alpha R_p(t - AT_{single}), & \text{if } \frac{m_{j-1}}{bw_{j-1}} \le t < \frac{m_j}{bw_j} \\ \dots & \\ \sum_{i=0}^{N-2} m_i + bw_{N-1} t - \alpha R_p(t - AT_{single}), & \text{if } \frac{m_{N-2}}{bw_{N-2}} \le t < \frac{m_{N-1}}{bw_{N-1}} \\ \sum_{i=0}^{N-1} m_i - \alpha R_p(t - AT_{single}), & \text{if } \frac{m_{N-1}}{bw_{N-1}} \le t \le (\frac{L}{R_p} + AT_{single}) \end{cases} \quad (4.1)$$

where AT_{single} is the access time using PWR single installment strategy and α is the parameter that controls the buffer occupancy either by flushing the buffer that is currently consumed ($\alpha = 1$) or by retaining the retrieved data without flushing until the end of presentation ($\alpha = 0$). The former case is typical of an applications such as *pay-per-view* kind of movie services and the latter is typical of an applications such as *interactive* movie viewing services on networks. Thus, any value of $0 \le \alpha \le 1$, is a measure of the extent to which interactivity

is provided by the service provider. Hence, depending on the current network and server loading conditions the service provider may vary the value of α for clients, thus exercising different levels of interactivity with the server systems. This may also be a measure of QoS and hence the pricing for users may be varied as per clients interactive requirements.

When $\sum_{i=0}^{N-1} bw_i \leq \alpha R_p$, the minimum buffer size expected of single installment retrieval strategy denoted as B_{single}^{min}, would be,

$$B_{single}^{min} = \sum_{i=0}^{N-1} \frac{bw_i}{bw_0} cs_0 \tag{4.2}$$

However, when $\sum_{i=0}^{N-1} bw_i > \alpha R_p$, B_{single} will increase until portion m_j has been totally retrieved from server S_j, where $\sum_{i=j}^{N-1} bw_i \geq \alpha R_p$ and $\sum_{i=j+1}^{N-1} bw_i < \alpha R_p$. Then, B_{single}^{min} would be

$$B_{single}^{min} = \sum_{i=0}^{j} m_i + \sum_{i=j+1}^{N-1} \left(\frac{bw_i}{bw_j} m_j \right) - \alpha R_p \left(\frac{m_j}{bw_j} - AT_{single} \right) \tag{4.3}$$

Similarly, the size of buffer space demanded by the PWR multi-installment strategy B_{multi}, is given by,

$$B_{multi} = \begin{cases} \sum_{i=0}^{N-1} bw_i t, & \text{if } 0 \le t < AT_{multi} \\ \sum_{i=0}^{N-1} bw_i t - \alpha R_p (t - AT_{multi}), & \text{if } AT_{multi} \le t < \frac{\sum_{i=0}^{n-1} m_{0,i}}{bw_0} \\ \sum_{i=0}^{n-1} m_{(0,i)} + \sum_{i=1}^{N-1} bw_i t - \alpha R_p (t - AT_{multi}), & \text{if } \frac{\sum_{i=0}^{n-1} m_{0,i}}{bw_0} \le t < \frac{\sum_{i=0}^{n-1} m_{1,i}}{bw_1} \\ \dots & \\ \sum_{j=0}^{k-1} \sum_{i=0}^{n-1} m_{j,i} + \sum_{i=k}^{N-1} bw_i t - \alpha R_p (t - AT_{multi}), & \text{if } \frac{\sum_{i=0}^{n-1} m_{k-1,i}}{bw_{k-1}} \le t < \frac{\sum_{i=0}^{n-1} m_{k,i}}{bw_k} \\ \dots & \\ \sum_{j=0}^{N-2} \sum_{i=0}^{n-1} m_{j,i} + bw_{N-1} t - \alpha R_p (t - AT_{multi}), & \text{if } \frac{\sum_{i=0}^{n-1} m_{N-2,i}}{bw_{N-2}} \le t < \frac{\sum_{i=0}^{n-1} m_{N-1,i}}{bw_{N-1}} \\ \sum_{j=0}^{N-1} \sum_{i=0}^{n-1} m_{j,i} - \alpha R_p (t - AT_{multi}), & \text{if } \frac{\sum_{i=0}^{n-1} m_{N-1,i}}{bw_{N-1}} \le t \le (\frac{L}{R_p} + AT_{multi}) \end{cases} \tag{4.4}$$

where AT_{multi} is the access time using PWR multi-installment strategy. Now, in the case of $\sum_{i=0}^{N-1} bw_i \le \alpha R_p$, the minimum of buffer size demanded by multi-installment retrieval strategy denoted as B_{mulit}^{min}, would be

$$B_{multi}^{min} = \sum_{i=0}^{N-1} \frac{bw_i}{bw_0} cs_{0,0} \tag{4.5}$$

In the case of $\sum_{i=0}^{N-1} bw_i > \alpha R_p$, B_{multi} will increase until portion $m_{j,n-1}$, the last portion from server S_j, has been totally retrieved, where $\sum_{i=j}^{N-1} bw_i \ge \alpha R_p$ and $\sum_{i=j+1}^{N-1} bw_i < \alpha R_p$. Then, B_{multi}^{min}

would be,

$$B_{multi}^{min} = \sum_{i=0}^{j}\sum_{k=0}^{n-1} m_{i,k} + \sum_{i=j+1}^{N-1}\left(\frac{bw_i}{bw_j}\sum_{k=0}^{n-1} m_{j,k}\right) - \alpha R_p \left(\frac{\sum_{k=0}^{n-1} m_{j,k}}{bw_j} - AT_{multi}\right) \tag{4.6}$$

4.2 Buffer Constraints

As a fixed-size buffer is preferred in any commercially available client machine, it will be wiser if the pre-allocated space during system initialization for this VoD application can be reused without further invoking memory allocation service from the operating system. Clearly, if the Operating System (OS) renders too little buffer this will cause an overflow and jitters during presentation at the client. On the other hand, adding buffers beyond a certain limit will not further improve system performance. As far as the performance of this strategy is concerned, an important question to address is as follows: *Given a buffer size of B bits at the client site, can we expect a continuous presentation by using our PWR multi-installment strategy?* To answer this question, we need to consider both the buffer occupancy and the access time. For the ease of analysis, we only consider the case of homogenous channels and set $\alpha = 1$. Note that $\alpha = 1$ implies that the system is of *pay-per-view* kind of service framework.

In order to compute the minimum buffer size expected at the client site, we need to determine exactly the time instants at which these servers finish transferring their last installments to the client. When $\sum_{i=0}^{N-1} bw_i < \alpha R_p$, from (3.53) and (4.5), we deduce

$$\lim_{n\to\infty} B_{multi}^{min} = \left(\frac{R_p - Nbw}{R_p}\right) L \tag{4.7}$$

However, when $\sum_{i=0}^{N-1} bw_i \geq \alpha R_p$, it is complex to determine these time instants at which these servers finish transferring their last portion of movie except for the last server. To clarify this aspect, we now consider an example that allows us to observe the relationships explicitly to derive these time instants at which these servers finish transferring their last installments of

the requested movie.

Example 1. Consider a scenario in which the requested movie is supplied by 3 servers, S_0, S_1 and S_2 by using our multi-installment strategy. From (3.26) we have,

$$\begin{aligned} m_{2,i} &= \sigma(m_{1,i} + m_{0,i}) \\ m_{1,i} &= \sigma(m_{0,i} + m_{2,i-1}) \\ m_{0,i} &= \sigma(m_{2,i-1} + m_{1,i-1}), \quad i = 0, 1, \ldots, n-1 \end{aligned} \tag{4.8}$$

Further, (4.8) can be rewritten as,

$$\begin{aligned} \sum_{i=0}^{n-1} m_{2,i} &= \sigma(\sum_{i=0}^{n-1} m_{1,i} + \sum_{i=0}^{n-1} m_{0,i}) \\ \sum_{i=0}^{n-1} m_{1,i} &= \sigma(\sum_{i=0}^{n-1} m_{0,i} + \sum_{i=0}^{n-2} m_{2,i}) \\ \sum_{i=0}^{n-1} m_{0,i} &= \sigma(\sum_{i=0}^{n-2} m_{2,i} + \sum_{i=0}^{n-2} m_{1,i}) \end{aligned} \tag{4.9}$$

Note that when we divide the left sides of each of the above expressions by the connection bandwidth bw, we can compute the time instants at which the last installment will be completed. Now, we have,

$$\frac{\sum_{i=0}^{n-1} m_{2,i}}{\sum_{i=0}^{n-1} m_{1,i}} = (1+\sigma) - \sigma \left(\frac{\sum_{i=0}^{n-2} m_{2,i}}{\sum_{i=0}^{n-1} m_{1,i}} \right) \tag{4.10}$$

Further,

$$\frac{\sum_{i=0}^{n-1} m_{1,i}}{\sum_{i=0}^{n-2} m_{2,i}} = \sigma(1 + \sigma \left(1 + \frac{\sum_{i=0}^{n-2} m_{1,i}}{\sum_{i=0}^{n-2} m_{2,i}} \right)) \tag{4.11}$$

When the number of installments tends to be large (infinity), $n \to \infty$, we realize that

$$\frac{\sum_{i=0}^{n-1} m_{k,i}}{\sum_{i=0}^{n-1} m_{j,i}} = \frac{\sum_{i=0}^{n-2} m_{k,i}}{\sum_{i=0}^{n-2} m_{j,i}} \tag{4.12}$$

This means that the ratio of the sizes of the loads between different servers remains more-or-less identical when a very large number of installments is considered. Thus, by using (4.11) and (4.12) in (4.10), we have,

$$\frac{\sum_{i=0}^{\infty} m_{2,i}}{\sum_{i=0}^{\infty} m_{1,i}} = (1+\sigma) - \left(1 + \sigma(1 + \frac{\sum_{i=0}^{\infty} m_{1,i}}{\sum_{i=0}^{\infty} m_{2,i}})\right)^{-1} \tag{4.13}$$

Similarly we have,

$$\frac{\sum_{i=0}^{\infty} m_{1,i}}{\sum_{i=0}^{\infty} m_{0,i}} = (1+\sigma) - \left(1 + \frac{\sum_{i=0}^{\infty} m_{2,i}}{\sum_{i=0}^{\infty} m_{1,i}}\right)^{-1} \tag{4.14}$$

From (4.13) and (4.14) we conclude,

$$\frac{\sum_{i=0}^{\infty} m_{2,i}}{\sum_{i=0}^{\infty} m_{1,i}} = \frac{\sum_{i=0}^{\infty} m_{1,i}}{\sum_{i=0}^{\infty} m_{0,i}} \tag{4.15}$$

Equation (4.15) means that the loads on these 3 servers, from S_0 to S_2, form a geometric progression.

In fact this holds true for using any number of servers. Thus, generalizing for N servers, we can rewrite (3.26) as,

$$\sum_{i=0}^{n-1} m_{k,i} = (\sum_{p=k+1}^{N-1} \sum_{i=0}^{n-2} m_{p,i} + \sum_{p=0}^{k-1} \sum_{i=0}^{n-1} m_{p,i})\sigma, \quad k = 0, 1, 2, \cdots, N-1 \tag{4.16}$$

Further, we have,

$$\sum_{i=0}^{n-1} m_{k,i} - \sum_{i=0}^{n-1} m_{k-1,i} = (\sum_{i=0}^{n-1} m_{k-1,i} - \sum_{i=0}^{n-2} m_{k,i})\sigma, \quad k = 0, 1, 2, \cdots, N-1 \tag{4.17}$$

Then, together with (4.12), we have:

$$\frac{\sum_{i=0}^{\infty} m_{k,i}}{\sum_{i=0}^{\infty} m_{k-1,i}} = (1+\sigma) - \left[\sum_{i=N-k-1}^{N-2} \sigma^{i-(N-k-1)} \left(\sum_{j=N-1-i}^{N-1} \frac{\sum_{i=0}^{\infty} m_{j,i}}{\sum_{i=0}^{\infty} m_{k,i}}\right)\right]^{-1}, \quad k = 1, 2, \cdots, N-1 \tag{4.18}$$

The above relationships are rather difficult to analytically verify the geometric progression followed by the ratios of the loads on the servers. However, we can immediately conform by direct simulation of the above expression to validate this claim in Chapter 5.

Now denoting the ratio between the loads on the adjacent servers as γ, we can express,

$$\sum_{i=0}^{\infty} m_{j,i} = \gamma^j \sum_{i=0}^{\infty} m_{0,i} \tag{4.19}$$

Further,

$$\sum_{i=0}^{\infty} m_{N-1,i} = \gamma^{N-1} \sum_{i=0}^{\infty} m_{0,i} = \frac{bwL}{R_p} \tag{4.20}$$

Thus solving equation (4.20) we obtain the value of γ. Together with (4.6), we have:

$$\lim_{n\to\infty} B_{multi}^{min} = \left(\sum_{i=0}^{N-M-1} \left(\frac{bw}{R_p} \gamma^{i-N+1} \right) - \frac{Mbw}{R_p} \gamma^{-M} - \gamma^{-M} \right) L \tag{4.21}$$

where,

$$M = \left\lfloor \frac{R_p}{bw} \right\rfloor \tag{4.22}$$

The minimum buffer size proposed in using PWR multi-installment strategy is derived under the assumption that the number of installments is very large. Thus, the B_{multi}^{min} obtained from either (4.7) or (4.21) is the lower bound of the buffer requirement at the client site. In reality when the number of installments is finite, the client will not have a smooth presentation with a buffer size less than B_{multi}^{min} obtained from either (4.7) or (4.21). We shall testify our findings through rigorous simulation tests later in Chapter 5.

Chapter 5

Simulations and Discussion

In this chapter, we shall evaluate the performance of our PWR single installment and multi-installment strategies. In our simulation experiments, we considered the case when the connection bandwidths are identical, i.e., $bw_i = bw$ for all the channels. The movie size L is assumed to be 2Gbits, and the playback rate R_p is 1.5Mbps.

5.1 Access Time

We first present the performance of our single installment strategy. As it is evident from the closed-form solution, the access time monotonically decreases as we tend to utilize more and more number of servers. Figure 5.1 shows this behavior of the access time with respect to the number of servers utilized. The connection bandwidth bw is 1Mbps. In the figure, we have shown the plots using both PWR single installment strategy and PAR single installment strategy, respectively. As expected, as the requested movie is available on more servers, the access time decreases. From these plots we observed that the PWR single installment strategy remarkably outperforms the PAR single installment strategy on minimizing the access time. Typically, when using 3 servers, the access time of PAR single installment strategy is 376.16 seconds. However, the access time of PWR single installment strategy is 52.51 seconds. Thus,

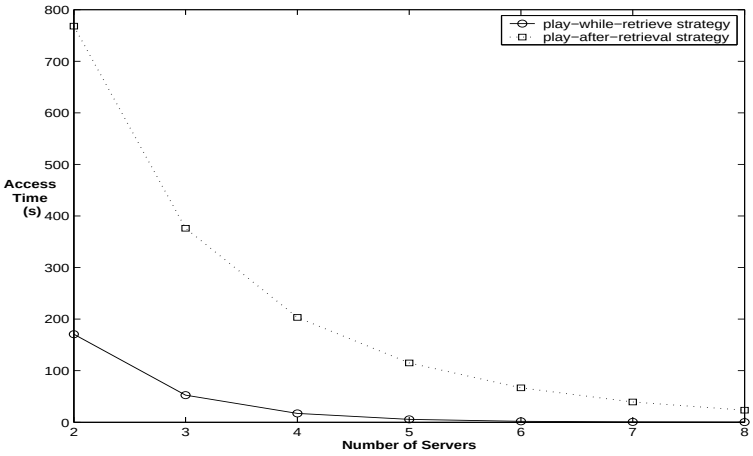

Figure 5.1: Access time vs number of servers using PWR and PAR: single installment strategy

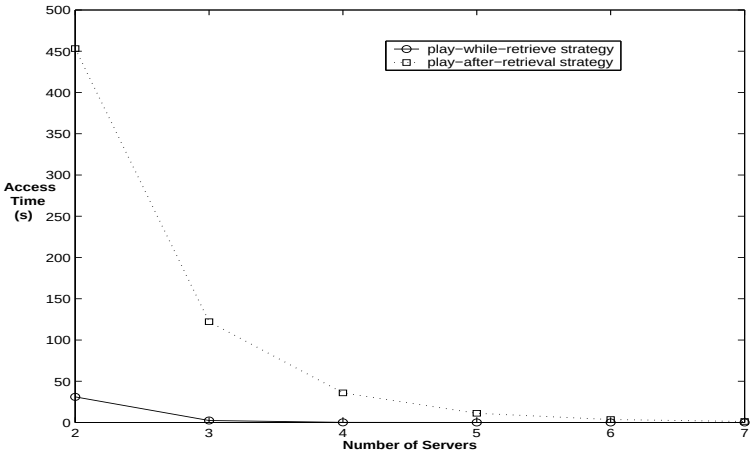

Figure 5.2: Access time vs number of servers with $n = 2$
using PWR and PAR: multi-installment strategy

we gain a significant decrease of 86.04% in this case.

In the case of multi-installment strategy, we have two parameters, the number of servers and the number of installments, to control the retrieval procedure. First, we see the influence of the number of servers on the access time. Figure 5.2 shows the behavior of the access time with respect to the number of servers, while in our simulation experiments the number of servers is varied from 2 onwards. The connection bandwidth bw is 1Mbps. In this figure, we have shown the plots using both PWR and PAR multi-installment strategies corresponding access

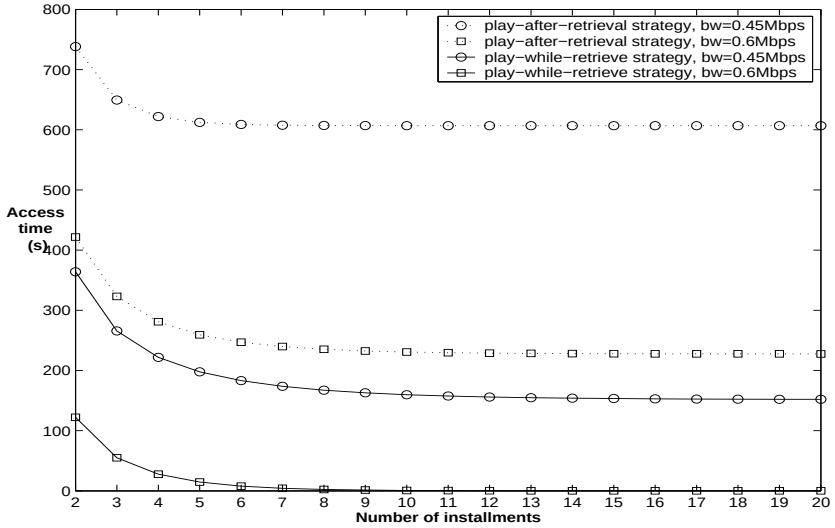

Figure 5.3: Access time vs number of installments using PWR and PAR: multi-installment strategy

times when $n = 2$. From these plots we observe that the PWR multi-installment strategy also outperforms the PAR multi-installment strategy. For example, when using 3 servers, the access time of PAR multi-installment strategy is 122.19 seconds. However, the access time of PWR multi-installment strategy is 2.42 seconds. Thus, we gain a striking reduction of 98.02%. Comparing the performance shown in Figure 5.2 with Figure 5.1, we observed that there is a significant reduction on the access between PWR multi-installment and single installment strategies. Typically, in the case of using 3 servers, we gained a reduction of 95.39%.

Now, we show the effect of the number of installments on the access time in Figure 5.3. The number of servers considered in this experiment is 3 while the number of installments is varied from 2 onwards. Further, in order to evaluate the effect of connection bandwidth on the access time, we used two different connection bandwidths, 0.45Mbps and 0.6Mbps, respectively. From our results, we observed that when the number of installments is increased the access time using both PWR and PAR multi-installment strategies tend to decrease at first. Then the access time using either strategy tends to quickly saturate to a value when the number of installments is increased indefinitely. Also, it may be observed that the saturation

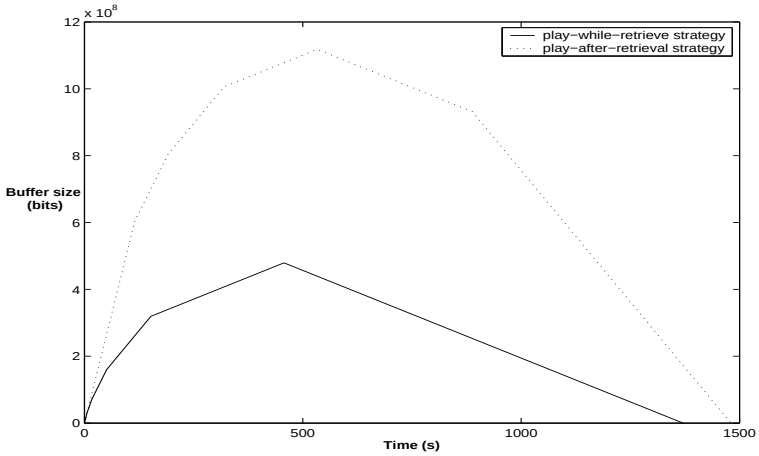

Figure 5.4: Client buffer occupancy using PWR and PAR: single installment strategy

of access time is quick in the case of PAR strategy when compared with the PWR strategy. The plots also reveal the fact that even with smaller connection bandwidths PWR strategy has a clear advantage of yielding a minimum access time when compared with PAR strategy with higher connection bandwidths. Further, using the PWR strategy, in the case of $Nbw \geq R_p$, the saturation value of access time is 0, otherwise, this value is given by (3.53). These observations also testify the results of our asymptotic analysis in Chapter 3.2.3.

5.2 Client Buffer Requirement

We now evaluate the buffer requirements at the client site. We set $\alpha = 1$. First, we considered the single installment retrieval strategy employing 5 servers S_0 to S_4 with connection bandwidths of 1Mbps. Figure 5.4 shows the behavior of buffer occupancies at the client site with respect to time t using both PWR and PAR single installment strategies. As expected, the buffer requirement using PWR single installment strategy is much less than that of the PAR single installment strategy. In our experiments, the maximum buffer space expected by the former strategy is 457Mbits and for the latter it is 1.041Gbits. Thus, we have a reduction of 57.13% on buffer requirement. The influence of α on buffer requirements can be demonstrated

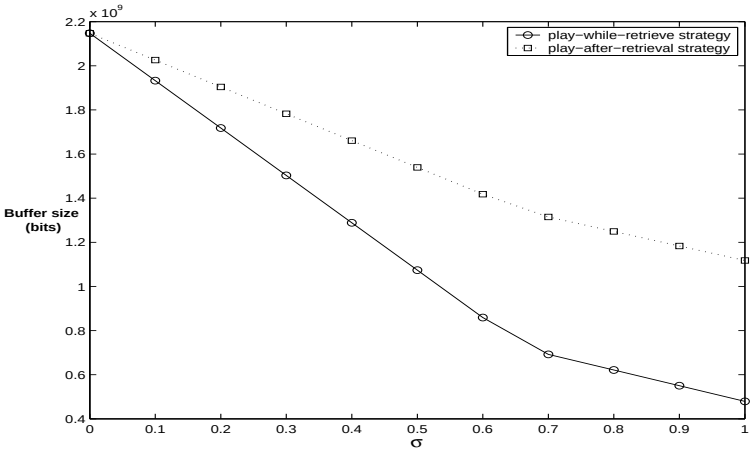

Figure 5.5: Client buffer size vs α using PWR and PAR: single installment strategy

by varying α in the range $[0, 1]$, as shown in Figure 5.5. Thus we observed that regardless of α value, the buffer requirement imposed by the PWR single installment strategy is smaller than that of the PAR single installment strategy. Typically, when $\sigma = 1$, we gain a significant reduction of 57.13% on the buffer requirement. Even at $\alpha = 0.5$, we also gained a reduction of 30.25% on the buffer requirements.

Now we consider the multi-installment retrieval strategy employing 5 servers S_0 to S_4 using connection bandwidths of 0.3Mbps and the number of installments of 4. The behavior of buffer requirement using multi-installment retrieval strategy is demonstrated by Figure 5.6. In this figure, we show the buffer occupancies at the client site with respect to time t using both PWR and PAR multi-installment strategies. From the results, the maximum buffer needed by PWR strategy is 215.6Mbits, while the maximum buffer needed by PAR strategy is 317.8Mbits. Thus, we have a decrease of 32.16% on the buffer requirement. While this behavior is somewhat identical to the single installment strategy, the behavior becomes interesting when the effect of connection bandwidth is also considered. We show the effect of connection bandwidth on buffer requirement in Figure 5.7. In this experiment, the connection bandwidth is varied from 0.1Mbps onwards. We observed that the buffer requirement reaches a minimum value at a point where the cumulative connection bandwidth becomes equal to

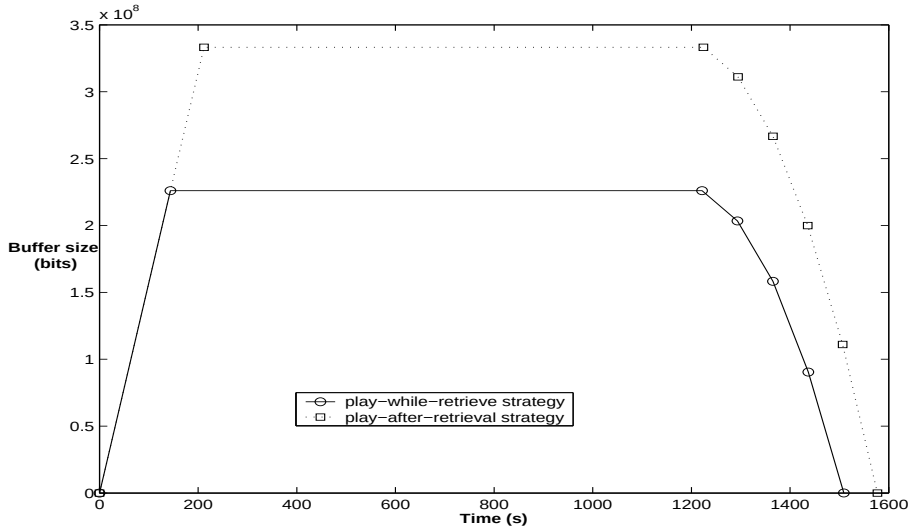

Figure 5.6: Client buffer occupancy using PWR and PAR: multi-installment strategy

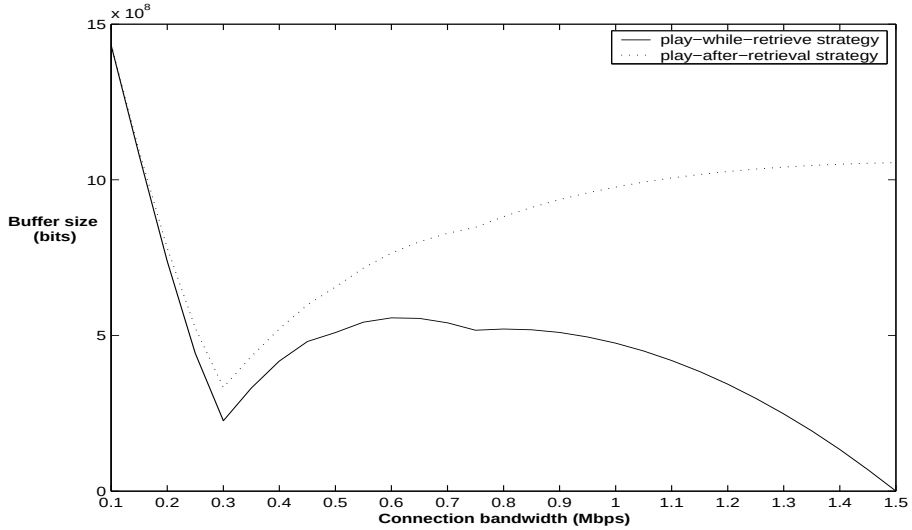

Figure 5.7: Client buffer occupancy vs connection bandwidth using PWR and PAR: multi-installment strategy

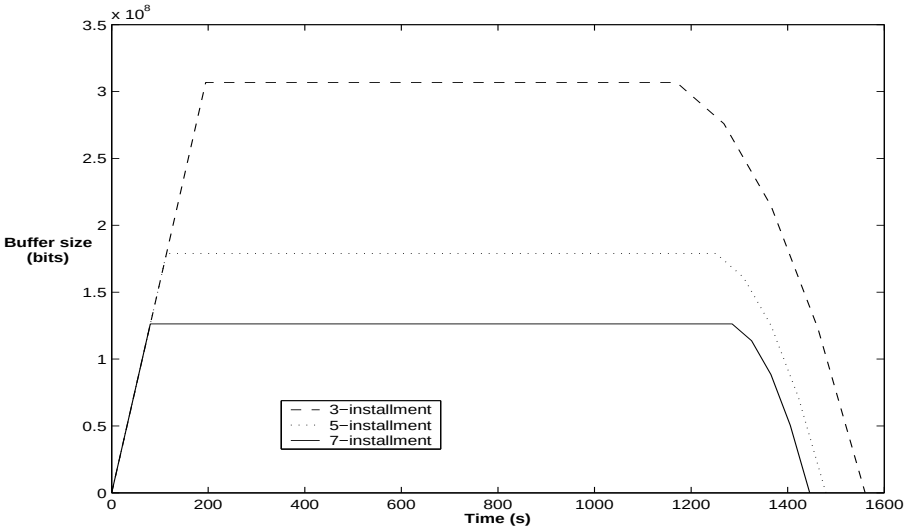

Figure 5.8: Client buffer occupancy vs number of installments using PWR multi-installment strategy

the playback rate of the movie. Afterwards, the buffer requirement trend seems to increase for both the strategies although PWR strategy clearly wins the race. Another observable effect is that for large connection bandwidth magnitudes, the buffer requirement tends to decrease faster for PWR while the requirement continues to increase in the case of another strategy. This is due to the fact that by using PWR multi-installment strategy, as the connection bandwidth approaches the playback rate, practically few data need to be stored and access time approaches zero.

Now, we show the effect of number of installments on buffer requirement on Figure 5.8 for PWR strategy. The number of servers considered is 5 with connection bandwidths of 0.3Mbps. Clearly, using a larger number of installments minimizes the buffer requirements at the client site. Typically, we obtained a reduction of 58.82% in buffer requirement between 3-installment strategy (292.6Mbits) and 7-installment strategy (120.5Mbits) are used.

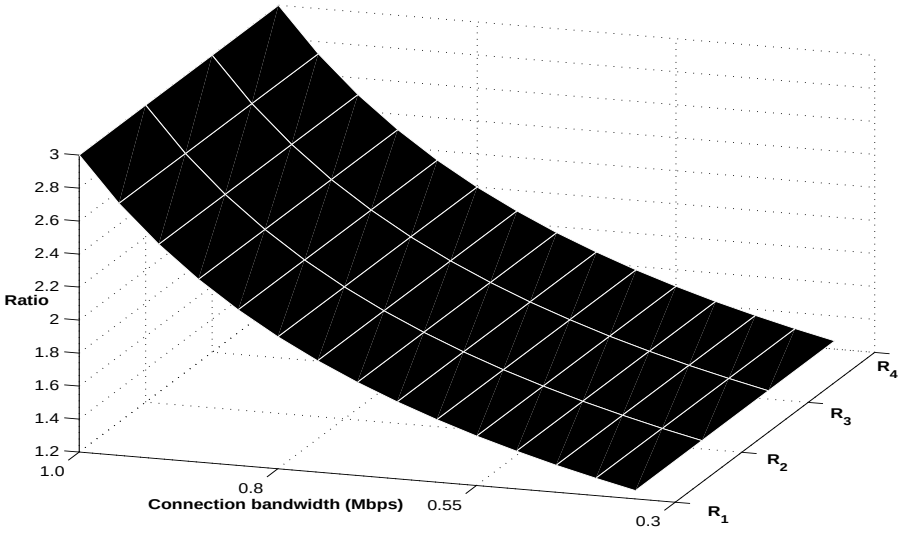

Figure 5.9: Loads on servers vs connection bandwidth using PWR single installment strategy

5.3 Load Balancing

From our earlier experiments, we observed that the loads (amount of portion(s) rendered by a server) on servers are not identical and hence it would be interesting to quantify the ratios of the amount of loads rendered by adjacent servers with respect to different connection bandwidths. First, we considered the PWR single installment strategy. We define the ratios as $R_i = m_i/m_{i-1},\ i = 1, ..., 4$. The connection bandwidth in this experiment is varied from 0.3Mbps onwards. Figure 5.9 shows the ratios of loads on adjacent servers using PWR single installment strategy. From our results, we observed that $R_i = R_j,\ \forall i \neq j$, for a given bandwidth. Also, the sizes of the portions rendered by different servers, S_0 to S_{N-1}, form a geometric progression.

Now, we consider the PWR multi-installment strategy. Figure 5.10 shows the ratios of loads on adjacent servers using PWR multi-installment strategy with connection bandwidths of 0.5Mbps. As in the previous experiment, we define the ratios as $R_i = \sum_{j=0}^{n-1} m_{i,j} / \sum_{j=0}^{n-1} m_{i-1,j}$, $i = 1, ..., 4$. As the number of installments increases, the ratios of loads rendered by adjacent

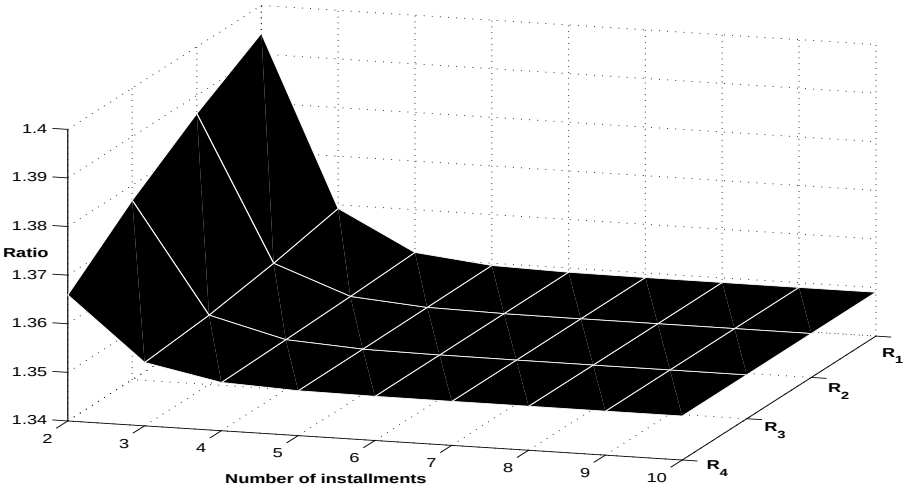

Figure 5.10: Loads on servers using PWR multi-installment strategy

servers become identical, i.e., $R_i = R_j,\ \forall i \neq j$, and even in this case, as testified by our analysis, we observed that the sizes of the portions rendered by different servers, S_0 to S_{N-1}, form a geometric progression.

From above experiment, we know that the ratios of loads rendered by adjacent servers become identical by using a relative large number of installments. It is natural to examine the impact of connection bandwidth on the ratio by simulation experiment. In this experiment, the connection bandwidth is varied from 0.1Mbps onwards. Figure 5.11 shows the ratios of loads on adjacent servers using PWR multi-installment strategy. We observed that the ratio remains at 1 when $Nbw \leq R_p$. Afterwards, the ratio increases when the connection bandwidth is increased. Further, comparing Figure 5.9 with Figure 5.11, we observed the ratio using PWR multi-installment strategy is a slightly smaller than that of using PWR single installment strategy, at a given connection bandwidth.

In the next chapter, we shall implement our PWR strategy in a real-life VoD system to evaluate the applicability of the strategy.

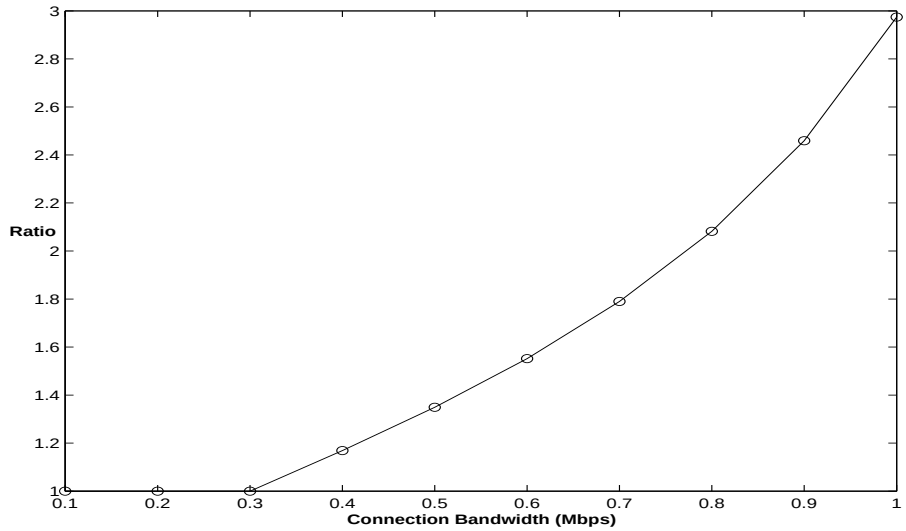

Figure 5.11: Loads on servers vs connection bandwidth using PWR multi-installment strategy

Chapter 6

Experiments in A Real-life VoD System

While the concept of utilizing geographically distributed servers to retrieve a long duration movie (in parts) is novel to the literature, no experimental evidence was presented to quantify the performance of the system. In this chapter, we report our experience in the implementation of our PWR strategy in a real-life VoD system, Jini VoD (JVoD) system [29, 30, 31], which is realized on the Ethernet network to utilize multiple servers. This rigorous and full-fledged implementation clearly justifies the applicability of the multiple servers retrieval strategy to real-life network based service infrastructure.

6.1 Introduction to the JVoD System

JVoD system is a real-life VoD system which utilizes multiple servers to render the distributed multimedia server to clients. The entire system is employing the recent Jini technology on a Java platform. Several intricate aspects of Jini technology are thoroughly exploited to realize a successful working system. The attractive feature of this system is in exploiting Jini's lookup service (JLS), a kind of lookup procedure that offers an elegant and flexible service. In the

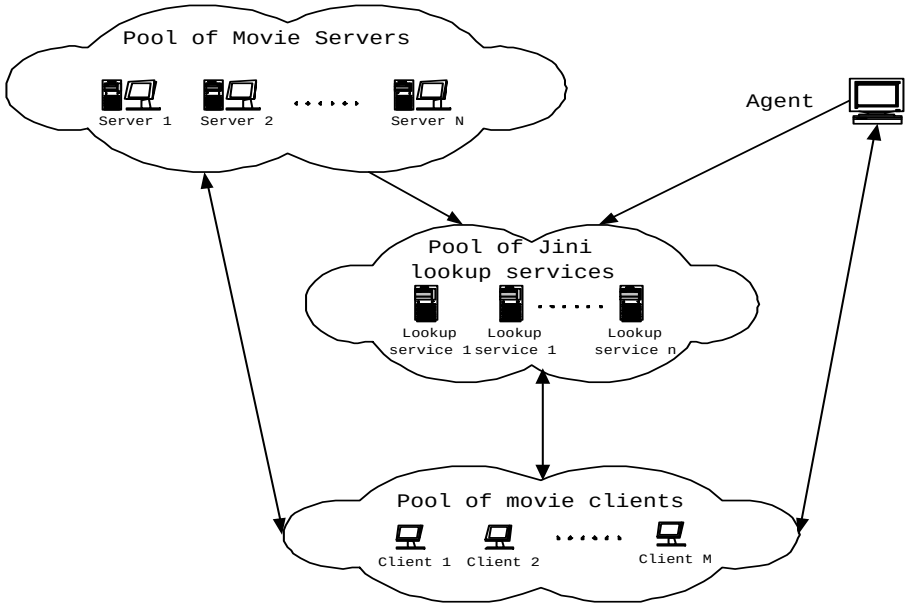

Figure 6.1: Overall view of the JVoD software architecture

system, an Agent is employed to coordinate the activities between the pool of Movie Servers and the pool of clients. In addition, code migration from this Agent to the clients is carried out for instructing the clients upon certain events. Further, the Movie Servers only send the specified data to the clients upon receiving the request of clients. Thus, the JVoD system is a complete Agent driven pull-based VoD system.

6.1.1 Components of the JVoD System

The interconnection medium in our actual implementation is an Ethernet LAN in which a set of processors will act as Movie Servers and a set of processors will act as clients. The basic architecture of this JVoD system is shown in Figure 6.1 and comprises the following components.

A. Movie Server: The Movie Server is basically the host that stores the actual movie files that can be retrieved and viewed by the client. Besides movie files, each Movie Server also maintains a small database that is used to record the transaction history, which will be used to trace the server activities.

B. Agent: The Agent is the "brain" of the entire JVoD system. It is solely responsible on how the clients should behave in different situations. This is achieved by a downloadable part of the Agent, referred to as a *rule-base*, which enables the client to make decisions. The rule-base basically decides on *how*, *where* and *which* are the Movie Servers to contact and to retrieve different portions of the movies. In our design, the Agent not only aids the clients to carry out the required (streaming) transaction under normal conditions, but also instructs the clients (during the retrieval process) on how to handle the exceptions that might arise due to unpredictable server behavior, problem of some missing files, etc.

C. JLS: JLS is very much similar to the *naming server* [32] used in other distributed network paradigms and in systems such as CORBA. It holds the registration information of all services (the Agent and the Movie Servers in our case) available in the system. More precisely, a JLS maps interfaces indicating the functionality provided by a service to sets of objects that implement the service. The JLS acts as an conciliator to a client who is looking for a service. Any client who needs to make use of a Jini service will first contact the JLS. Then, the intended Service Object [33] is subsequently transferred from the JLS to the requesting client site where it will be used to set up the connection between the client and the Jini service. The Service Object contains the Java programming language interface for the service, including the methods that service consumers will invoke to execute the service along with any other descriptive attributes. Once the connection is established, the JLS is not involved in any of the subsequent interactions between that client and that service.

D. Client: The client is an application featuring a player. Further, through the preview screen, users can select and preview movies before they decide to view the entire movie. To fully utilize the multiple Movie Servers in the system, the client is allowed to receive concurrent (incoming) streams.

Thus, the above four essential components comprise the entire working JVoD system. Below,

we shall explain on how a client typically interacts with the system, as an overview.

6.1.2 Interaction of A Client with Other Components

The client in the JVoD system is *dumb* in the sense that it cannot make any decision regarding how the movie files are to be accessed from the Movie Servers registered with the JLS. After the client application is initiated, it downloads the rule-base from the Agent on-the-fly, then it can contact the respective Movie Servers, by using the strategies inside this rule-base. Exceptions that could arise are also handled by this rule-base. It would be meaningful to look on what exactly happens during a client transaction.

Assuming that the JLS and the Agent service are initiated first in the system. The components such as the clients and Movie Servers can become a part of the system anytime, since the Jini framework can self-recover as long as the JLS is available. Assuming that a new client application has been initiated, the client would first register itself with the JLS and obtain the necessary information regarding the Movie Servers from the JLS. However, the client will need to authenticate itself with the Agent before it can download the rule-base. Once the authentication requirements of the client are fulfilled (like entering the correct username and password), the client is said to be "logged on" to the current Agent, which will provide the algorithms(rule-base) that the client may use.

Once the user passes the authentication phase, he/she can then use the client application to open a comprehensive Movie Chooser that features a list of all the movies that are currently available (similar to a directory of list of available movies or yellow pages, with movie information) and a small preview screen. With the Movie Chooser opened, the user can select any available movie and proceed to either view the trailer of the selected movie or to "buy" and view the entire movie immediately. When the user prefers to preview the trailer first, the client application requires to download the *Server Locating* (SL) strategy from the Agent.

This SL strategy aids the client to locate a Movie Server from which the movie trailer can be retrieved. Our current SL strategy adopts a policy in which the client will select a Movie Server with a larger available bandwidth (depending on the current network and server loading conditions, the service provider can change the policy without involving the client), to retrieve the movie trailer. Once the trailer has been retrieved, it will be played in the preview screen. Further, once the SL strategy is transferred from the Agent to the client, the client need not download it from the Agent again until the client application is closed and restarted later. This means the client can reuse the strategy in its subsequent transactions (in a single session) till it quits the system.

After viewing some trailers, the user may then decide whether to buy the movies and view it. Once the user decides to view the entire movie, the client application needs to download more algorithms(strategies) from the Agent in order to fulfill this transaction. The strategies to be downloaded include, the *Streaming* strategy, *Buffer Management* strategy, *Scheduling and Retrieval* strategy, *Emergency Server Generating* strategy, etc. With these strategies, the client will proceed to start retrieving data streams from Movie Servers. Since this system supports multiple servers, the movie will most probably be streamed from more than one server. For example, if there are three Movie Servers hosting the selected movie, the algorithms from the Agent most likely will instruct the client application to stream the movie data from these three servers using separate connections. This implies that the movie data will be partitioned into three portions and streamed from each of these three servers. Once the connections between the client and servers were established, the movie data will be retrieved by the client and the buffering technique will handle these multiple incoming streams. After the critical size of the first portion data had been received, the client application will start the playback of the movie.

As mentioned above, during the retrieval process, a number of critical situations may occur. Among these, there are two major critical situations including unpredictable failure of Movie

Servers and failure of the Agent. The failure of Movie Servers definitely has a major impact on the performance of the system. Considering the case when there are three Movie Servers and one of them crashed during streaming movie data to the client, obviously, the client will not be able to stream the scheduled portion of the movie from that failed server. In this case, the Emergency Server Generating algorithm that was transferred from the Agent apriori will handle such situation for the client. A *backup Movie Server* (one of the other two Movie Servers hosting the requested movie will act as a backup Movie Server) would be generated for the client to contact and resume streaming of the lost portion from the time instant where it stopped.

All these mechanisms are transparent to the user and the playback of the movie will not be affected, unless the playback at the client site is in a *race* with the data streaming. The possibility of this racing is extremely low, as the algorithm transferred from the Agent had calculated the best possible critical size which ensures that the playback of the movie would circumvent this racing with the data streaming. However, it would be catastrophic if all the Movie Servers participating in the retrieval process were crashed before all portions were retrieved completely. If this worst scenario happens, the user would be notified that the playback of this movie would be affected. However, the user can continue viewing the available portions of this movie, or can go back to the Movie Chooser and select another available movie to view. On the other hand, the crashing of the Agent has no effect on the playback of the movie. This is because the Agent is not involved in the transmission of movie file after the connections between the client and the Movie Servers have been set up. However, in this case, the user would be notified that the Agent had exited the system and hence, it would not be possible to view other movies unless the client application is re-authenticated with the same or another Agent.

6.2 Experiments of the PWR Strategy on the JVoD System

In this section, we shall conduct experiments to demonstrate the performance of our PWR strategy, more precisely, the single-installment PWR strategy. The experiments are run with Pentium series dedicated PCs with Windows 98 OS or Windows 2000 OS, on a 100Mbps Ethernet LAN platform. Two computers are configured exclusively to participate as Movie Servers. One of the computers is configured to participate as a JVoD Agent. Finally, three computers in our system are configured to participate as clients. Each Movie Server, Agent and client has a 128MB RAM and a hard-disk of capacity 20GB. Each client is also a Pentium machine equipped with a JMF player (a built in player within the JVoD client). Each Movie Server is capable of storing 15 movies (typically of 110 minutes duration) together with the respective trailers for previewing.

6.2.1 Retrieval Process

In our JVoD system, once the JLS, the Movie Servers and the Agent are ready, the client application can be launched. After launching a similar interface as shown in Figure 6.2 is displayed at the client site. From the status bar at the bottom of client interface, we can see that there are 2 Movie Servers and 1 Agent in this running JVoD system. After the user logs on to the system (enter the correct username and password), he/she can browse the information of the desired movie from the Movie Chooser and preview the trailer of movie, as shown in Figure 6.3.

The retrieval processing begins when the user decides to “buy” the desired movie. First, the bandwidth test is conducted between all available Movie Servers and this client. Then, the movie portions can be retrieved from the servers according to the results generated by the ESP retrieval strategy. The status of retrieval processing can be monitored from the interface

Figure 6.2: Accessing JVoD service through the client application - screen shot of the client

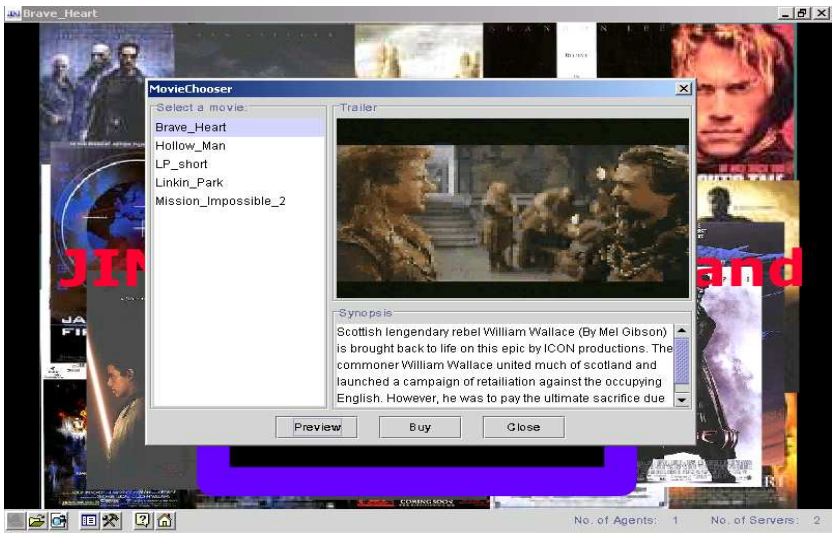

Figure 6.3: Choosing and previewing the movie trailer - screen shot of the client

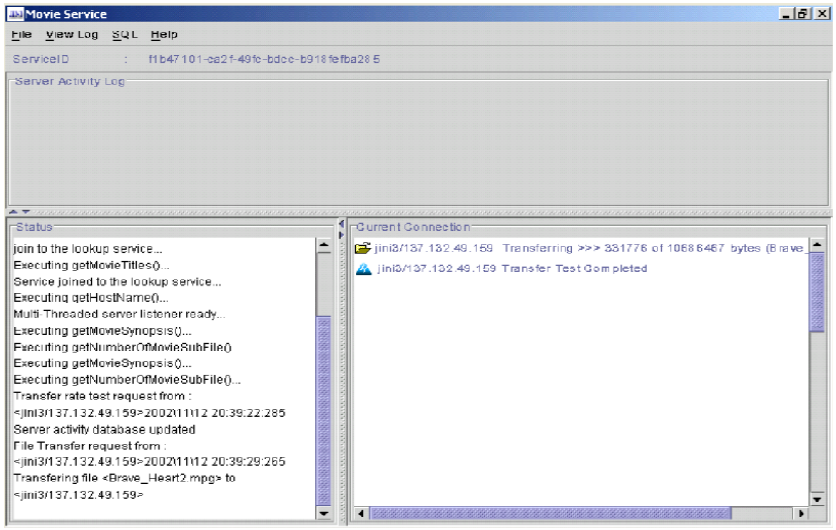

Figure 6.4: Retrieving the movie portions from Movie Servers - screen shot of the Movie Server

of all participating Movie Servers, as shown in Figure 6.4. Without knowing such background activities, what the user needs to do is just to click the mouse once,

The presentation will begin at the client site shown in Figure 6.5 after the critical size has been retrieved, which is calculated by the PWR strategy. Even with one Movie Servers crashed during the retrieval process, the lost portion can be retrieved from the remaining Movie Server, and the presentation will not be interrupted. After the presentation comes to the end or the user terminates it, the Agent will calculate the corresponding bill and update the database of users.

6.2.2 Access Time

An important performance metric in the experiments is the *access time*. It may be noted that the access time depends on several parameters, such as the size of the movie file, bandwidth of the connection channel, server capacity or the response time of the servers, etc. In the experiments, since the network infrastructure is somewhat dedicated, the access time will be mainly measured with respect to the size of the movies and the bandwidth, which have been

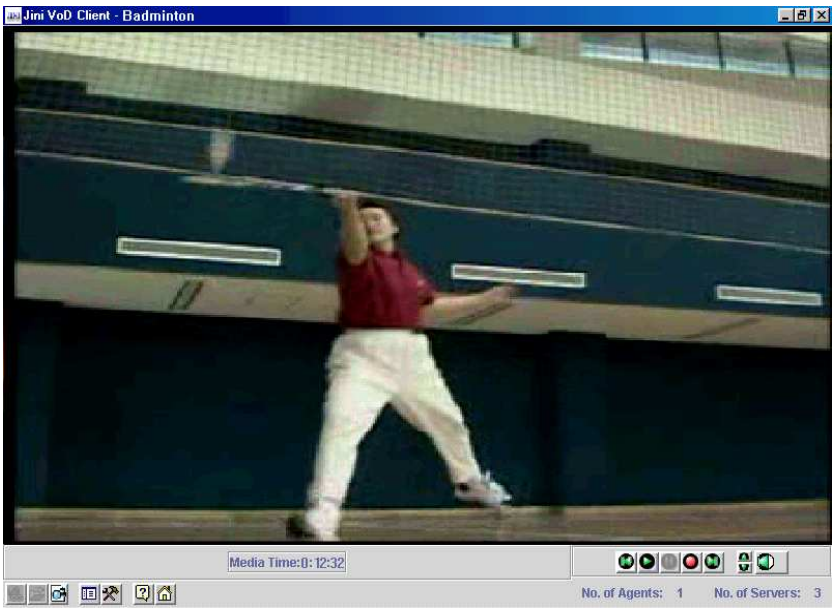

Figure 6.5: Presentation at the client site - screen shot of the client

carefully discussed in Chapter 3.1. We conducted the experiments with 3 MPEG1 movies of different sizes and 2 different bandwidths of the server-client channel. The playback rate of such files is about 1.36Mbps. To obtain an accurate result, we used the average access time over 20 times experiments for each movie retrieval.

Figure 6.6 shows this behavior of the access time with respect to the number of servers utilized, and the bandwidth of connection channel is 1.29Mbps. As expected, as the requested movie is available on more Movie Servers, the access time decreases. The access time increases when the movie size increases.

Figure 6.7 shows the behavior of the access time with respect to the bandwidth of the connection channel utilized. As expected, as the bandwidth of the connection channel increases, the access time decreases.

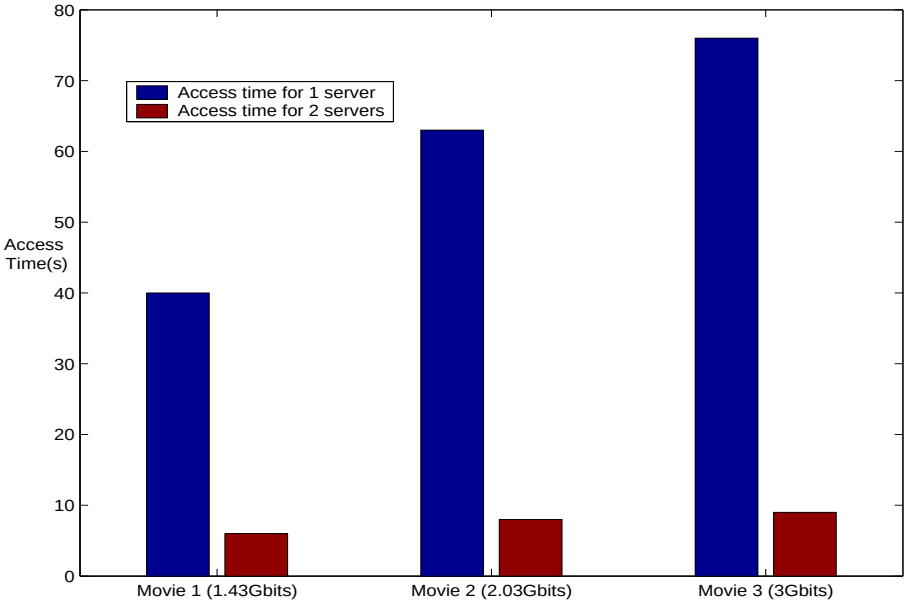

Figure 6.6: Access Time vs number of Movie Servers

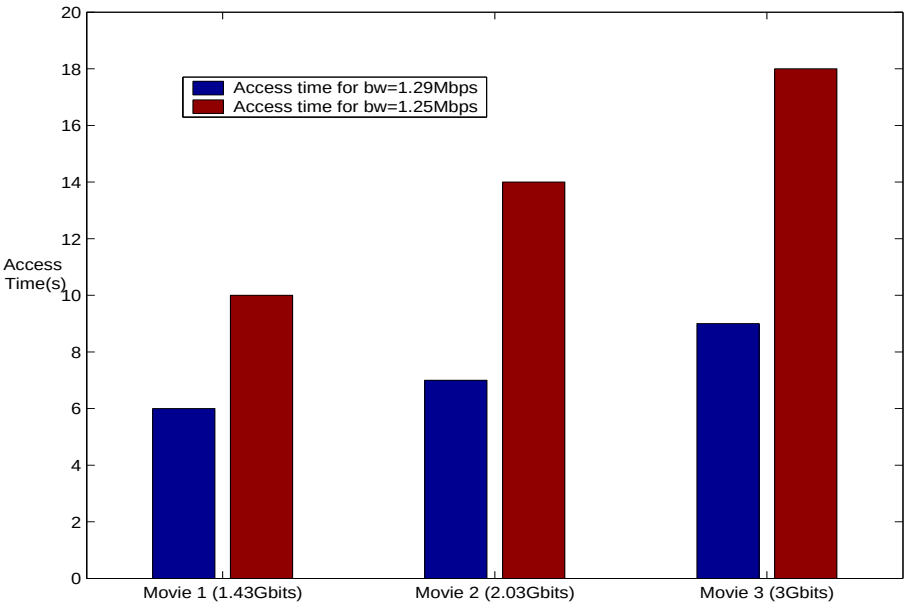

Figure 6.7: Access Time vs bandwidth of the connection channel

Chapter 7

Conclusions and Future Work

In this thesis, we have presented a generalized approach to the theory of retrieving a long-duration movie requested by a client using a network based multimedia service infrastructure. For a network based environment, we have designed and analyzed an efficient PWR playback strategy to minimize the access time of the movie. Our analysis clearly highlights the advantages of the strategy when compared to a PAR approach. Further, with our design, we have shown that both the playback strategies (PWR and PAR) can in turn choose to retrieve the movie portions using either single installment or multi-installment retrieval strategies. Thus playback strategies are basically concerned about *how* and *when* to initiate the playback while retrieval strategies are concerned about *how* to retrieve the movie data from the servers. The use of PWR or PAR depends on the application requirements. For instance, for a pay-per-view kind of multimedia service, PWR is suitable as it is also shown to expect a minimum buffer requirement, given by (4.7) or (4.21), at the client site. However, when an interactive service is to be provided, PAR is the natural choice, however, client is expected to have a bigger buffer size than using PWR strategy, as shown in our simulation experiments. Further, for the pay-per-view service with PWR strategy, depending on server availabilities, one may tune the number of installments to be used to suit buffer availability at the client site. This is clearly evident from our experiments (Figure 5.8). Of course, when the availability of servers

is somewhat constrained in the system, then using multi-installment strategy may not be possible. In such situations, single installment strategy may be meaningful, however at the cost of a larger buffer space consumption and a longer access time.

For PWR strategy, we have derived closed-form solutions for the access times using both single installment and multi-installment retrieval strategies. Further, we have derived a closed-form expression for the *critical size* that must be first retrieved to kick-start the movie presentation. Also, for our PWR playback strategy employing multi-installment retrieval strategy, we have conducted a rigorous asymptotic performance analysis. Our asymptotic analysis elicits performance bounds on the strategies and serves as valuable measures to tune the system performance. For instance, together with our simulation experiments, the choice on the minimum number of installments can be made depending on the saturation level of the access time. Also, the choice on the number of servers to be utilized for a given number of installments can also be quantified.

Our analysis, followed by experimental support, clearly renders clues on buffer management at the client site and also on tuning the server system to meet the buffer availability at the client site. Thus, service facility can be attractive to clients even with low buffer space, by tuning the number of installments.

Our implementation of PWR playback strategy in the JVoD system testifies the applicability of such strategy in the real-life VoD system. Our implementation demonstrated that our strategy can achieve the goal to minimize the access time of the client while providing a highly fault-tolerant distributed VoD service. This design and implementation study will certainly benefit Internet service providers who wish to render an attractive VoD services on networks.

Finally, following are some interesting issues that can be considered as open-ended problems within the context of the problem addressed in this thesis.

1. What happens on a server failure? Typically, how the missing load can be retrieved?

2. How does the server system respond when the client interacts with the presentation in the case of PAR service?

3. How to handle multiple client requests using PAR and PWR strategies?

It would be interesting to address the above issues to realize a practically working multiple servers VoD system.

Bibliography

[1] D. Ghose and H-J. Kim, "Scheduling Video Streams in Video-on-Demand Systems: A Survey", *Multimedia Tools and Applications*, Vol. 11, No. 2, pp. 167-195, 2000.

[2] T.C. Kwok, "Residential broadband internet services and applications requirements, *IEEE Communications Magazine*, Vol. 35, No. 6, pp. 76C83, 1997.

[3] J.Y.B. Lee, "Parallel Video Servers: A Tutorial", *IEEE Multimedia*, Vol. 5, No. 2, pp. 20-28, April-June 1998.

[4] B. Veeravalli and G.D. Barlas, "Access Time Minimization for Distributed Multimedia Applications," *Multimedia Tools and Applications*, Kluwer Academic Publishers, Vol. 12, No. 2/3, pp. 235-256, Nov. 2000.

[5] C. Bernhardt and E. Biersack, "A Scalable Video Server: Architecture, Design and Implementation," *Proceedings of the Realtime Systems Conference, 1995.*, Paris, France, pp. 63-72, January 1995.

[6] J.Y.B. Lee, "Concurrent Push - A Scheduling Algorithm for Push-Based Parallel Video Servers", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 3, Apr. 1999.

[7] J.Y.B. Lee and P.C. Wong, "Performance Analysis of a Pull-Based Parallel Video Server", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 12, pp. 1217-1231, Dec. 2000.

[8] Lougher, P. Pegler, D. Shepherd, D., "Scalable storage servers for digital audio and video," *International Conference on Storage and Recording Systems, 1994.*, pp. 140-143, 1994.

[9] J.Y.B. Lee and P.C Wong, "server array approach for video-on-demand service on local area networks", *Proceedings of INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies*, Vol. 1, pp. 27-34, March 1996.

[10] Li-gang Dong, B. Veeravalli, and C.C. Ko, "Efficient Movie Retrieval Strategies for Movie-on-Demand Multimedia Services on Distributed Networks" (To appear in *Multimedia Tools and Applications*, Kluwer Academic, 2003). Also available as a technical report: # TRMM-1/VB/Opensource/2000-1, Open Source Software Laboratory, *The National University of Singapore*, Singapore (http://opensource.nus.edu.sg/∼elebv).

[11] C.C. Bisdikian, and B.V. Patel, "Cost-Based Program Allocation for Distributed Multimedia-on-Demand Systems", *IEEE Multimedia*, pp. 62-72, Fall issue, 1996.

[12] H.M. Vin, A. Goyal and P. Goyal, "Algorithms for Designing Large-Scale Multimedia Servers", *Computer Communications*, Vol. 18, No. 3, pp. 192-203, Mar. 1995.

[13] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting," *Multimedia Systems*, Vol. 4, pp. 197-208, 1996.

[14] C.C. Aggarwal, J.L. Wolf and P.S. Yu, "Design and Analysis of Permutation-Based Pyramid Broadcasting," *Multimedia system*, Vol. 7, pp. 439-448, 1999.

[15] Won Y. and Srivastava J., "Strategic Replication of Video Files in a Distributed Environment", *Multimedia Tools and Applications*, Vol. 8, No. 1, pp. 249-283, Mar. 1999.

[16] C. Papadimitriou, S. Ramanathan, P.Venkat Rangan and S. Sampath Kumar, "Multimedia Information Caching for Personalized Video-on-Demand", *Computer Communications*, Vol. 18, No. 3, pp. 204-216, Mar. 1995.

[17] B. Ping, B. Prabhakaran, and A. Srinivasan, "Retrieval Scheduling for Collaborative Multimedia Presentations", *Multimedia Systems*, ACM/Springer-Verlag, No. 8, pp. 146-155, 2000.

[18] Y. Wang, J.C.L. Liu, D.H.C. Du, and J. Hsieh, "Efficient Video File Allocation Schemes for Video-on-Demand Services," *Multimedia System*, No. 5, pp. 283-296, 1997.

[19] A. Srivastava, A. Kumar, and A. Singru, "Design and Analysis of a Vide-on-Demand Server," *Multimedia Systems*, ACM/Springer-Verlag, Vol. 5, pp. 238-254, 1997.

[20] H.H. Pang, B. Jose, and M.S. Krishnan, "Resource Scheduling in a High-Performance Multimedia Server," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 2, March/April 1999.

[21] S. Lau and J.C.S. Lui, "Scheduling and Data Layout Policies for a Near-Line Multimedia Storage Architecture," *Multimedia System*, No. 5, pp. 310-323, 1997.

[22] D. Jadav, A.N. Choudhary, and P.B. Berra, "Techniques for Increasing the Stream Capacity of a High-Performance Multimedia Server," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 2, March/April 1999.

[23] K.A. Hua, and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Cannes France, Sep. 1997

[24] Y. Won, and J. Srivastava, "SMDP: Minimizing Buffer Requirements for Continuous Media Servers", *Multimedia Systems*, ACM/Springer-Verlag, No. 8, pp. 105-117, 2000.

[25] P.V. Rangan, and H.M. Vin, "Efficient Storage Techniques for Digital Continuous Multimedia", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 4, pp. 564-573, 1993.

[26] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz, "A Disk-Based Storage Architecture for Movie on Demand Servers", *Information Systems*, Vol. 20, No. 6, pp. 465-482, 1995.

[27] A.Dan and D.Sitaram, "Multimedia Caching Strategies for Heterogeneous Application and Server Environments", *Multimedia Tools and Applications*, Kluwer Academic, Vol. 4, No. 3, pp. 279-312, May. 1997.

[28] H. Fahmi, S. Baqai, A. Bashandy, and A. Ghafoor, "Dynamic Resource Allocation for Multimedia Document Retrieval over High Speed LANs," *Multimedia Tools and Applications*, Vol. 8, No. 1, pp. 91-114, 1999.

[29] See Ying Lai, "Jini Technology Support for Efficient Video/Movie-on-Demand Multimedia: Overall Framework Design with Jini Technology," *B.Eng Thesis*, National Universiyt of Singapore, 2001

[30] Hun Yen Kwoon, "Jini Technology Support for Efficient Video/Movie-on-Demand Multimedia: Stream Handling and Time Minimization Strategy," *B.Eng Thesis*, National Universiyt of Singapore, 2001

[31] Goh Kar Whee, "Jini Technology Support for Effecient Video/Movie-on-Demand Multimedia System: Implemenation of JMF and Buffering Strategy," *B.Eng Thesis*, National Universiyt of Singapore, 2001

[32] Andrew S. Tanenbaum, Robbert Van Renesse, "Distributed operating systems," *ACM Computing Surveys (CSUR)*, Vol. 17, Issue. 4, pp. 419 - 470, December 1985.

[33] Sun Microsystems Inc. "Jini Technology Core Platform Specification," *http://wwws.sun.com/software/jini/specs/jini1.1html/core-title.html*